

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-344105

(P2001-344105A)

(43) 公開日 平成13年12月14日 (2001. 12. 14)

(51) Int.Cl.⁷

G 0 6 F 9/44

識別記号

F I

G 0 6 F 9/06

テーマコード* (参考)

6 2 0 A 5 B 0 7 6

6 2 0 H

審査請求 未請求 請求項の数20 O L (全 26 頁)

(21) 出願番号 特願2001-6006 (P2001-6006)

(22) 出願日 平成13年1月15日 (2001. 1. 15)

(31) 優先権主張番号 特願2000-97395 (P2000-97395)

(32) 優先日 平成12年3月31日 (2000. 3. 31)

(33) 優先権主張国 日本 (J P)

(71) 出願人 000233055

日立ソフトウェアエンジニアリング株式会
社

神奈川県横浜市中区尾上町6丁目81番地

(72) 発明者 細美 彰宏

神奈川県横浜市中区尾上町6丁目81番地

日立ソフトウェアエンジニアリング株式会
社内

(74) 代理人 100096954

弁理士 矢島 保夫

最終頁に続く

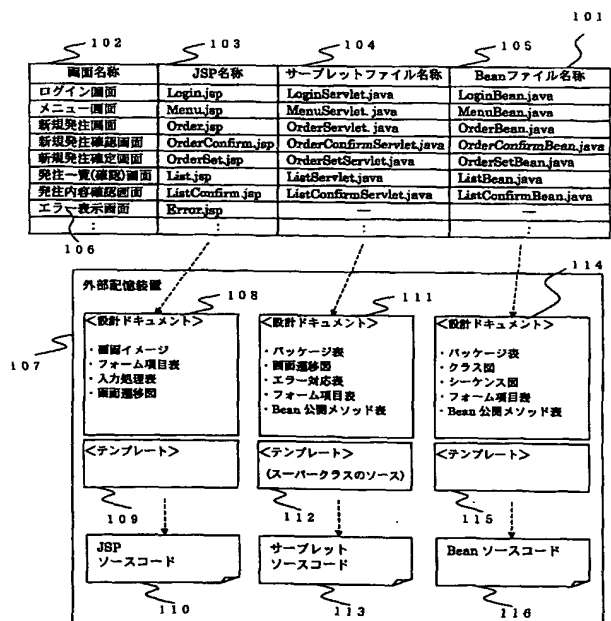
(54) 【発明の名称】 Webアプリケーション開発方法、開発支援システム、および該方法に係るプログラムを記憶した記憶媒体

(57) 【要約】

【課題】 Web上に新規サービスを早期に追加したり、サービスを拡張したりするために、Webアプリケーション・システムを短期間に開発することが求められている。開発作業を軽減させ、スピードアップを図り、かつ、複雑なシステムや機能拡張などに対応するためにも、各コンポーネントの対応を明確にし、それぞれの役割を分離して設計する必要がある。

【解決手段】 Webアプリケーション・システムの開発において、それぞれの画面に対応させて、サーブレット、JSP、およびBeanの各コンポーネントを1対1の関係で定義し、各コンポーネントの対応を明確にする。画面デザインから各コンポーネントの定義を対応表として生成する対応表生成部を持ち、その対応表と設計ドキュメントの情報を利用して、サーブレット、JSP、およびBeanのコードを自動生成するコード自動生成部とによって、開発作業を単純にし、軽減することができる。

対応表



【特許請求の範囲】

【請求項 1】 インターネットやイントラネット上で、サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント (Bean) による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発方法であって、

前記Webアプリケーションシステムの設計仕様にある画面仕様に基づいて、画面毎に対応するサーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称を対応させて定義することによって、サーブレット、JSP、およびBeanの開発を行うことを特徴とするWebアプリケーション開発方法。

【請求項 2】 インターネットやイントラネット上で、サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント (Bean) による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発方法であって、

前記Webアプリケーションシステムの設計仕様にある画面仕様に基づいて、画面毎に対応するサーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称を 1 対 1 に対応させて定義することによって、サーブレット、JSP、およびBeanの開発を行うことを特徴とするWebアプリケーション開発方法。

【請求項 3】 請求項 1 または 2 に記載のWebアプリケーション開発方法において、前記画面と前記サーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称との対応関係と、前記設計仕様の情報を利用して、各コンポーネントのソースコードを自動生成することを特徴とするWebアプリケーション開発方法。

【請求項 4】 請求項 1 または 2 に記載のWebアプリケーション開発方法において、前記Beanのコンポーネントは、HTMLページを表示するために必要な情報をすべて持ち、HTMLページの情報とデータベースから取得する情報とをマッピングする役割を持つことを特徴とするWebアプリケーション開発方法。

【請求項 5】 請求項 3 に記載のWebアプリケーション開発方法において、前記サーブレット、JSP、およびBeanの各コンポーネントのソースコードの自動生成の際、生成するコンポーネント毎に、テンプレート一覧を表示しその中から選択されたテンプレートを雛形とし、該雛形に前記設計仕様に応じたコードを埋め込むことによりソースコードを自動生成することを特徴とするWebアプリケーション開発方法。

【請求項 6】 インターネットやイントラネット上で、サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント (Bean) による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発方法であって、

設計仕様にある画面仕様を読み込み、各画面毎に、画面名称に対応させたサーブレット、JSP、および/またはBeanの各コンポーネントの名称を生成するステップと、生成した名称の各コンポーネントについて、そのコンポーネントのソースファイルの雛形となるテンプレートをテンプレート一覧から選択させるステップと、そのコンポーネントの設計仕様に基づくコードを前記テンプレートに埋め込むことにより、そのコンポーネントのソースコードを自動生成するステップとを備えたことを特徴とするWebアプリケーション開発方法。

【請求項 7】 請求項 6 に記載のWebアプリケーション開発方法において、前記テンプレート一覧は、スーパークラスのテンプレートを含むものであり、前記ソースコードを自動生成するステップは、該スーパークラスを継承してソースコードを作成するものであることを特徴とするWebアプリケーション開発方法。

【請求項 8】 請求項 6 に記載のWebアプリケーション開発方法において、前記各コンポーネントの名称を生成するステップは、前記各画面毎に、画面名称に 1 対 1 対応で、サーブレット、JSP、および/またはBeanの各コンポーネントの名称を生成するものであることを特徴とするWebアプリケーション開発方法。

【請求項 9】 請求項 1 から 8 の何れか 1 つに記載のWebアプリケーション開発方法において、開発すべきWebアプリケーションシステムのうち、サーブレットはWebブラウザからのリクエストを受ける役割、Beanはサーブレットの要求に従い処理を行いその処理結果を保持する役割、JSPはBeanの処理結果を参照してWebブラウザ上に表示するHTMLを出力する役割、というように役割を分離したことを特徴とするWebアプリケーション開発方法。

【請求項 10】 請求項 1 から 9 の何れか 1 つに記載のWebアプリケーション開発方法に係るプログラム。

【請求項 11】 請求項 1 から 9 の何れか 1 つに記載のWebアプリケーション開発方法に係るプログラムを記憶した記憶媒体。

【請求項 12】 インターネットやイントラネット上で、サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント (Bean) による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発システムであって、前記Webアプリケーションシステムの設計仕様にある画面仕様に基づいて、画面毎に対応するサーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称を対応させて定義することによって、サーブレット、JSP、およびBeanの開発を行う手段を備えたことを特徴とするWebアプリケーション開発システム。

【請求項 13】 インターネットやイントラネット上で、

サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント（Bean）による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発システムであって、

前記Webアプリケーションシステムの設計仕様にある画面仕様に基づいて、画面毎に対応するサーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称を1対1に対応させて定義することによって、サーブレット、JSP、およびBeanの開発を行う手段を備えたことを特徴とするWebアプリケーション開発システム。

【請求項14】請求項12または13に記載のWebアプリケーション開発システムにおいて、

前記画面と前記サーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称との対応関係と、前記設計仕様の情報を利用して、各コンポーネントのソースコードを自動生成する手段を、さらに備えたことを特徴とするWebアプリケーション開発システム。

【請求項15】請求項12または13に記載のWebアプリケーション開発システムにおいて、前記Beanのコンポーネントは、HTMLページを表示するために必要な情報をすべて持ち、HTMLページの情報とデータベースから取得する情報とをマッピングする役割を持つことを特徴とするWebアプリケーション開発システム。

【請求項16】請求項14に記載のWebアプリケーション開発システムにおいて、

前記サーブレット、JSP、およびBeanの各コンポーネントのソースコードの自動生成の際、生成するコンポーネント毎に、テンプレート一覧を表示しその中から選択されたテンプレートを雛形とし、該雛形に前記設計仕様に応じたコードを埋め込むことによりソースコードを自動生成する手段を、さらに備えたことを特徴とするWebアプリケーション開発システム。

【請求項17】インターネットやイントラネット上で、サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント（Bean）による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発システムであって、

設計仕様にある画面仕様を読み込み、各画面毎に、画面名称に対応させたサーブレット、JSP、および／またはBeanの各コンポーネントの名称を生成する手段と、生成した名称の各コンポーネントについて、そのコンポーネントのソースファイルの雛形となるテンプレートをテンプレート一覧から選択させる手段と、そのコンポーネントの設計仕様に基づくコードを前記テンプレートに埋め込むことにより、そのコンポーネントのソースコードを自動生成する手段とを備えたことを特徴とするWebアプリケーション開発システム。

【請求項18】請求項17に記載のWebアプリケーション開発システムにおいて、

前記テンプレート一覧は、スーパークラスのテンプレートを含むものであり、前記ソースコードを自動生成する手段は、該スーパークラスを継承してソースコードを作成するものであることを特徴とするWebアプリケーション開発システム。

【請求項19】請求項17に記載のWebアプリケーション開発システムにおいて、

前記各コンポーネントの名称を生成する手段は、前記各画面毎に、画面名称に1対1対応で、サーブレット、JSP、および／またはBeanの各コンポーネントの名称を生成するものであることを特徴とするWebアプリケーション開発システム。

【請求項20】請求項12から19の何れか1つに記載のWebアプリケーション開発システムにおいて、

開発すべきWebアプリケーションシステムのうち、サーブレットはWebブラウザからのリクエストを受ける役割、Beanはサーブレットの要求に従い処理を行いその処理結果を保持する役割、JSPはBeanの処理結果を参照してWebブラウザ上に表示するHTMLを出力する役割、というように役割を分離したことを特徴とするWebアプリケーション開発システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、Webアプリケーション開発方法および開発支援システムに関し、特にサーバサイドJavaの技術を利用したWebアプリケーションシステムにおけるサーブレット、JSP、およびBeanの設計および開発支援の技術に関する。

【0002】

【従来の技術】近年、インターネット、イントラネット上のWebアプリケーションシステムで、サーバサイドJavaの技術を利用したシステムが普及してきている。従来のWebシステムでは、CGI（Common Gateway Interface）プログラムによるものが一般的であった。しかし、CGIに代わり、サーブレットを利用したシステム開発が主流になりつつある。サーブレットとは、CGIとよく似た機能を提供する技術で、クライアント（Webブラウザ）から送信されるリクエストをサーバ上で処理し結果を返送するサーバサイドのプログラムである。

【0003】最も基本的なサーブレットを利用したシステムにおいては、次のような使い方となる。Webブラウザから送信したリクエストに応じてサーブレットが起動され、そのサーブレットでデータベースへのアクセスを行ったり、そのアクセス結果を受けてデータ加工などの処理を行う。また、リクエストの内容に応じて分岐して処理したり、他のサーブレットを呼び出すなどして、リクエストの結果としてWebブラウザに返送する。サーブレットに代わって、JSPを利用する場合もある。JSP（JavaServer Pages）とはHTMLファイル中にJavaのコードを記述して、動的にページを生成することができる技術で

あり、サーブレットとJSPとを組み合わせる利用することもできる。

【0004】従来のサーブレット、JSP、およびBeanを利用したシステムの構成において、Beanはデータベースにアクセスするとともに、業務処理も行い、また、その情報を保持する役割を持っており、Beanの設計や開発にはサーブレット、JSPの開発以上にJavaの知識と経験が必要であった。また、従来の開発では、サーバサイドにおける各コンポーネント単位に共通化を行っていたために、サーブレット、JSP、またはそれらから利用するBean単位に分担して作業をしていた。そのためにはウォーターフォール式の開発に見られるように、開発対象があらかじめ明確に定まっている必要があった。

【0005】

【発明が解決しようとする課題】近年のシステム開発において、開発サイクルはますます短くなってきている。特に業務系システムや業務サービスをWeb上で公開する場合、その開発サイクルを短縮することは顧客ニーズとして強い要求事項である。Webアプリケーションシステムにおいて、Webページのデザインや情報を定期的に変更するだけでなく、そのシステム上で新規サービスを早期に公開したり、サービスを迅速に拡張したりと、短期間に開発を行うことが求められている。それらの顧客ニーズに対応し、Webアプリケーションシステムの開発サイクルに対応するためにも、その開発作業を軽減させ、かつ、スピードアップを図る必要がある。

【0006】また、従来の開発方法では、サーブレットやJSPでリクエストに応じた処理を行い、その処理結果に応じて画面を生成するプログラムを記述していた。そのため、システムが大規模になり、システムやプログラミングが複雑になるに従って、個々の開発者には広範囲なJavaの知識が要求され、開発を行うJavaプログラマを集める必要もあった。さらに、開発するサーブレットやJSPで行う処理の把握やコーディング、デバッグ、メンテナンス作業においても負担がかかっていた。

【0007】作業分担をサーブレット毎、サーブレットから利用するBean毎という単位で行う場合、それらを結合して動作を確認するためには、開発者からクラスが提供されるのを待つ必要があった。一方、分担開発する場合、サーブレット、JSP、およびBeanを担当者毎に独自に開発するため、共通化を図ることができない。そのため、開発作業は各担当者のスキルに依存し、メンテナンス作業も難しくなっていた。

【0008】本発明は、上述した課題を解決することを目的とするものである。すなわち、サーバサイドJavaの技術を利用したWebアプリケーションシステムにおけるサーブレット、JSP、およびBeanの設計開発において、設計開発作業の負担を軽減させ、かつ設計開発のスピードアップを図ることを目的とするものである。

【0009】

【課題を解決するための手段】上記目的を達成するため、請求項1に係る発明は、インターネットやイントラネット上で、サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント (Bean) による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発方法であって、前記Webアプリケーションシステムの設計仕様にある画面仕様に基づいて、画面毎に対応するサーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称を対応させて定義することによって、サーブレット、JSP、およびBeanの開発を行うことを特徴とする。

【0010】請求項2に係る発明は、インターネットやイントラネット上で、サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント (Bean) による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発方法であって、前記Webアプリケーションシステムの設計仕様にある画面仕様に基づいて、画面毎に対応するサーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称を1対1に対応させて定義することによって、サーブレット、JSP、およびBeanの開発を行うことを特徴とする。

【0011】請求項3に係る発明は、請求項1または2において、前記画面と前記サーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称との対応関係と、前記設計仕様の情報を利用して、各コンポーネントのソースコードを自動生成することを特徴とする。

【0012】請求項4に係る発明は、請求項1または2において、前記Beanのコンポーネントは、HTMLページを表示するために必要な情報をすべて持ち、HTMLページの情報とデータベースから取得する情報とをマッピングする役割を持つことを特徴とする。

【0013】請求項5に係る発明は、請求項3において、前記サーブレット、JSP、およびBeanの各コンポーネントのソースコードの自動生成の際、生成するコンポーネント毎に、テンプレート一覧を表示しその中から選択されたテンプレートを雛形とし、該雛形に前記設計仕様に応じたコードを埋め込むことによりソースコードを自動生成することを特徴とする。

【0014】請求項6に係る発明は、インターネットやイントラネット上で、サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント (Bean) による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発方法であって、設計仕様にある画面仕様を読み込み、各画面毎に、画面名称に対応させたサーブレット、JSP、および/またはBeanの各コンポーネントの名称を生成するステップと、生成した名称の各コンポーネントについて、そのコンポーネントのソースファイルの雛形となるテンプレートをテンプレート一覧から選択させるステップと、そのコンポーネントの設計仕様に基づくコードを前記テンプレートに

埋め込むことにより、そのコンポーネントのソースコードを自動生成するステップとを備えたことを特徴とする。

【0015】請求項7に係る発明は、請求項6において、前記テンプレート一覧は、スーパークラスのテンプレートを含むものであり、前記ソースコードを自動生成するステップは、該スーパークラスを継承してソースコードを作成するものであることを特徴とする。

【0016】請求項8に係る発明は、請求項6において、前記各コンポーネントの名称を生成するステップは、前記各画面毎に、画面名称に1対1対応で、サーブレット、JSP、および/またはBeanの各コンポーネントの名称を生成するものであることを特徴とする。

【0017】請求項9に係る発明は、請求項1から8において、開発すべきWebアプリケーションシステムのうち、サーブレットはWebブラウザからのリクエストを受ける役割、Beanはサーブレットの要求に従い処理を行いその処理結果を保持する役割、JSPはBeanの処理結果を参照してWebブラウザ上に表示するHTMLを出力する役割、というように役割を分離したことを特徴とする。

【0018】請求項10に係る発明は、請求項1から9の何れか1つに記載のWebアプリケーション開発方法に係るプログラムである。請求項11に係る発明は、請求項1から9の何れか1つに記載のWebアプリケーション開発方法に係るプログラムを記憶した記憶媒体である。

【0019】請求項12に係る発明は、インターネットやイントラネット上で、サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント (Bean) による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発システムであって、前記Webアプリケーションシステムの設計仕様にある画面仕様に基づいて、画面毎に対応するサーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称を対応させて定義することによって、サーブレット、JSP、およびBeanの開発を行う手段を備えたことを特徴とする。

【0020】請求項13に係る発明は、インターネットやイントラネット上で、サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント

(Bean) による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発システムであって、前記Webアプリケーションシステムの設計仕様にある画面仕様に基づいて、画面毎に対応するサーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称を1対1に対応させて定義することによって、サーブレット、JSP、およびBeanの開発を行う手段を備えたことを特徴とする。

【0021】請求項14に係る発明は、請求項12または13において、前記画面と前記サーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称と

の対応関係と、前記設計仕様の情報を利用して、各コンポーネントのソースコードを自動生成する手段を、さらに備えたことを特徴とする。

【0022】請求項15に係る発明は、請求項12または13において、前記Beanのコンポーネントは、HTMLページを表示するために必要な情報をすべて持ち、HTMLページの情報とデータベースから取得する情報とをマッピングする役割を持つことを特徴とする。

【0023】請求項16に係る発明は、請求項14において、前記サーブレット、JSP、およびBeanの各コンポーネントのソースコードの自動生成の際、生成するコンポーネント毎に、テンプレート一覧を表示しその中から選択されたテンプレートを雛形とし、該雛形に前記設計仕様に応じたコードを埋め込むことによりソースコードを自動生成する手段を、さらに備えたことを特徴とする。

【0024】請求項17に係る発明は、インターネットやイントラネット上で、サーバサイドJavaの技術であるサーブレット、JSP、およびJavaBeansコンポーネント (Bean) による構成を持つWebアプリケーションシステムを開発するWebアプリケーション開発システムであって、設計仕様にある画面仕様を読み込み、各画面毎に、画面名称に対応させたサーブレット、JSP、および/またはBeanの各コンポーネントの名称を生成する手段と、生成した名称の各コンポーネントについて、そのコンポーネントのソースファイルの雛形となるテンプレートをテンプレート一覧から選択させる手段と、そのコンポーネントの設計仕様に基づくコードを前記テンプレートに埋め込むことにより、そのコンポーネントのソースコードを自動生成する手段とを備えたことを特徴とする。

【0025】請求項18に係る発明は、請求項17において、前記テンプレート一覧は、スーパークラスのテンプレートを含むものであり、前記ソースコードを自動生成する手段は、該スーパークラスを継承してソースコードを作成するものであることを特徴とする。

【0026】請求項19に係る発明は、請求項17において、前記各コンポーネントの名称を生成する手段は、前記各画面毎に、画面名称に1対1対応で、サーブレット、JSP、および/またはBeanの各コンポーネントの名称を生成するものであることを特徴とする。

【0027】請求項20に係る発明は、請求項12から19において、開発すべきWebアプリケーションシステムのうち、サーブレットはWebブラウザからのリクエストを受ける役割、Beanはサーブレットの要求に従い処理を行いその処理結果を保持する役割、JSPはBeanの処理結果を参照してWebブラウザ上に表示するHTMLを出力する役割、というように役割を分離したことを特徴とする。

【0028】すなわち本発明は、例えば、設計ドキュメントにある画面仕様の各画面に対応させて、サーブレッ

ト、JSP、およびBeanの各コンポーネントのソースファイル名称を1対1に対応させて表形式で定義する対応表生成部と、その対応表と設計ドキュメントの情報を利用して、サーブレット、JSP、およびBeanの各コンポーネントの名称や変数、メソッドの定義、および各コンポーネント間の呼び出し関係を自動的に抽出し、ソースコードとして出力するコード自動生成部とを備えることを特徴とする。コード自動生成部によりソースコードを生成する際に使用する、各コンポーネントのテンプレートとなるコードを格納したテンプレートデータや、サーブレットのスーパークラスとなるソースコード、および生成された各コンポーネントのソースコードは、所定の外部記憶装置に記憶する。

【0029】このように、Webアプリケーションシステムの開発において、各画面に対してサーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称を1対1に対応させて定義することで、各コンポーネントの対応を明確にすることができる。また、サーブレット、JSP、およびBeanの処理の標準化を行うことにより、画面単位に開発作業を分担することもできる。

【0030】また、画面毎の各コンポーネントの対応を明確にし、それぞれの役割を分離して設計することで、複雑なシステムやシステムへの機能拡張などメンテナンスにも簡単に対応できる。サーバサイドJavaの技術を利用すれば、サーブレットはWebブラウザからのリクエストを受ける役割、Beanはサーブレットの要求に従い処理し、その処理結果を保持する役割、JSPはBeanの処理結果を参照してWebブラウザ上表示するHTMLを出力する役割、というように役割を明確に分離することができる。特にBeanに関しては、業務分析、設計の結果得られる業務に関するクラスやデータベースにアクセスするクラスなどを利用して、HTMLページを表示するために必要な情報をすべて保持するように設計することを特徴とする。

【0031】

【発明の実施の形態】以下、本発明を実施する場合の一形態について図面を参照して具体的に説明する。

【0032】本発明の第1の実施の形態について説明する。ここでは、図3で後述するサーブレット、JSP、Beanを利用したWebアプリケーションシステムを設計開発する例で説明する。特に、本発明は、設計ドキュメントにある画面仕様の各画面に対応させて、サーブレット、JSP、およびBeanの各コンポーネントのソースファイル名称を1対1に対応させて対応表として定義することを特徴としている。この対応表と設計ドキュメントの情報を活用して、サーブレット、JSP、およびBeanの各コンポーネントの名称や変数、メソッドの定義、および各コンポーネント間の呼び出し関係を自動的に抽出し、ソースコードを自動生成する。

【0033】図1は、本実施の形態において設計開発するWebアプリケーションシステムの画面とサーブレッ

ト、JSP、およびBeanとの対応表の一例を示す図である。対応表101は、画面名称102とJSP名称103とサーブレットファイル名称104とBeanファイル名称105とから成り、それぞれ画面に対して1対1となるようにファイル名称を定義する。JSP名称103の拡張子は“.jsp”、サーブレットファイル名称104とBeanファイル名称の拡張子は“.java”とする。例えば、画面名称“新規発注画面”に対しては、JSP名称103は“Order.jsp”とし、サーブレットファイル名称104は、JSP名称103に“Servlet”を付加した“OrderServlet.java”とする。同様に、Beanファイル名称105は、“Bean”を付加した“OrderBean.java”とする。“エラー表示画面”106の場合、サーブレットおよびBeanと1対1で対応するものではなく、いくつかのサーブレットから呼び出される画面であり、JSP名称103“Error.jsp”のみを定義してある。

【0034】図1中、107は様々な情報を格納する外部記憶装置を示す。外部記憶装置107には対応表101に定義したサーブレット、JSP、およびBeanの各コンポーネントのソースコード110、113、116を生成するために必要な情報を格納している。108および109はJSPソースコード110を生成するために必要な情報を示し、画面イメージ、フォーム項目表、入力処理表、および画面遷移図を記述した設計ドキュメント108とテンプレートデータ109から成る。111および112はサーブレットソースコード113を生成するために必要な情報を示し、パッケージ表、画面遷移図、エラー対応表、フォーム項目表、およびBean公開メソッド表を記述した設計ドキュメント111とテンプレートデータ112とサーブレット・スーパークラスソースコードデータから成る。114と115はBeanソースコード116を生成するために必要な情報を示し、パッケージ表、クラス図、シーケンス図、フォーム項目表、およびBean公開メソッド表を記述した設計ドキュメント114とテンプレートデータ115から成る。

【0035】図2は、Webアプリケーションシステムを設計開発するシステムのシステム構成の一例を示した図である。図2中、201はコマンド入力やマウスを使った操作を行う端末、202は本発明を実現する機能やデータを格納する中央処理装置、203は様々な情報を格納する外部記憶装置を示す。外部記憶装置203に格納されている設計ドキュメント204は、Webアプリケーションシステムの開発において、顧客からの要求や業務仕様から得られる画面仕様などの設計ドキュメント、および、画面デザインや各画面イメージを表すHTMLファイルなどが格納されている。図1中、108から116で示した設計ドキュメント、テンプレートデータ、および生成されたソースコードは、この外部記憶装置203に格納される。また、サーブレット・スーパークラスソースコードデータ212が格納される。

【0036】中央処理装置202に格納されている機能は、設計ドキュメント204に格納されている画面仕様か

ら画面情報を読み込み、画面とサーブレット、JSP、およびBeanとの対応表206を生成する対応表生成部205と、対応表206と画面仕様やシステム設計書などの設計ドキュメント204の情報とテンプレートデータ208とを利用して、サーブレット、JSP、およびBeanの各ソースコードを自動生成するコード自動生成部207とから構成される。テンプレートデータ208には、ソースコードを生成する際の各コンポーネントのテンプレートとして利用するほか、サーブレットのスーパークラスとなるソースコードを格納する。コード自動生成部207によって生成される各ソースコードは、サーブレットソースコードデータ209、JSPソースコードデータ210、およびBeanソースコードデータ211に、それぞれ格納される。

【0037】図3は、本実施形態のシステムのコード自動生成部207によって生成したサーブレット、JSP、およびBeanを利用したWebアプリケーションシステムの全体構成の一例を示した図である。このWebアプリケーションシステムは、クライアント301からWebブラウザ302によって利用される。Webブラウザ302上には、HTMLページ303が表示される。Webブラウザ302からのリクエストはHTTP経由でサーバ304側に送信される。サーバ304にはHTTPサーバ305とアプリケーションサーバ306があり、アプリケーションサーバ306にはサーブレットやJSPの実行環境が含まれている。

【0038】コード自動生成部207によって生成したサーブレット、JSP、およびBeanは、それぞれ実行可能な状態で、サーバ304側のアプリケーションサーバ306上に配置する。サーバ304側にリクエストが送信されると、リクエストに応じて該当するサーブレット307が起動される。サーブレット307は、対応するBean 308に処理を要求する。さらに、Bean 308はDBアクセスオブジェクト310を利用してデータベース311にアクセスし、業務に関する処理やデータ加工などを行い、その処理結果を保持する。DBアクセスオブジェクト310は、業務を分析、設計した結果、作成されるクラスや、再利用可能な既存のクラスである。次に、サーブレット307からJSP309にBean 308を渡し、JSP309を呼び出す。JSP309は、Bean 308で保持する処理結果を参照して、HTMLページを生成し、クライアント301側に返送する。返送されたHTMLページは、Webブラウザ302上に表示される。ここで、サーブレット307、JSP309、およびBean 308は、Webブラウザ302に表示するHTMLページページ303に対して1対1で定義してある。

【0039】サーブレット307は、Webブラウザ302からのリクエストの受け付け、Bean 308への処理要求、およびJSP309の呼び出しという、それぞれの間をつなぎ、制御する役割を持つ。JSP309はHTMLページを出力するための表示に関することを受け持ち、Bean 308は業務に関する処理を受け持つ。Bean 308やJSP309を利用せず、サーブレット307のみでWebブラウザ302からのリクエストを

処理することは可能であるが、Bean 308やJSP309を利用することでそれぞれの役割を明確に分離することで、機能分担を簡潔に把握することができる。

【0040】図4を参照して、対応表生成部205によって対応表101を生成する処理の流れを説明する。まず、設計ドキュメント204に格納されている画面仕様からすべての画面情報を読み込む（ステップ401）。画面情報から画面名称と画面コード名称を取得する（ステップ402）。画面コード名称は、例えば、各画面イメージとして作成されているHTMLファイルのファイル名称から取得してもいいし、画面情報として定義してあってもよい。また、端末201から各画面毎に入力してもよい。ここでは、取得した画面名称を“新規発注画面”、画面コード名称を“Order”として説明する。次に、取得した画面コード名称に拡張子“.jsp”を付けてJSP名称とする（ステップ403）。画面コード名称に“Servlet”を付加し、拡張子“.java”を付けてサーブレットファイル名称とする（ステップ404）。画面コード名称に“Bean”を付加し、拡張子“.java”を付けてBeanファイル名称とする（ステップ405）。ここで、各ファイル名称は“Order.jsp”、“OrderServlet.java”、“OrderBean.java”と定義される。画面名称と、定義されたJSP名称、サーブレットファイル名称、およびBeanファイル名称の各ファイル名称を対応表101の該当個所に設定する（ステップ406）。ステップ401で取得した画面情報のすべてに対して処理したかをチェックし（ステップ407）、すべて処理するまで、ステップ402からステップ406までの処理を繰り返す。すべての画面情報に対して処理を行ったら、最後に、対応表101を出力して終了する（ステップ408）。ここで生成された対応表101は、設計ドキュメント204に格納されてもよい。

【0041】次に、図5を参照して、サーブレット、JSP、およびBeanの各ソースコードをコード自動生成部207によって自動生成する処理の流れを説明する。まず、対応表生成部205によって生成した対応表101を読み込む（ステップ501）。対応表101から画面名称102を取得する（ステップ502）。ここで取得した画面名称を“新規発注画面”として説明する。取得した画面名称に対応するJSP名称103、サーブレットファイル名称104、およびBeanファイル名称105の各名称がすべて設定されているかチェックする（ステップ503）。例えば、図1中の“エラー表示画面”106の場合、JSP名称103“Error.jsp”のみであるため、ステップ503から502に戻る。各名称がすべて設定してある場合、以降の処理を行う。ステップ502で取得した画面である“新規発注画面”に対するサーブレットのソースコード“OrderServlet.java”を生成する（ステップ504）。同様に、JSPのソースコード“Order.jsp”を生成し（ステップ505）、Beanのソースコード“OrderBean.java”を生成する（ステップ506）。ステップ501で読み込んだ対応表101のすべての画面に対して処理したか

をチェックし(ステップ507)、すべて処理するまで、ステップ502からステップ506までの処理を繰り返す。すべての画面に対して処理を行ったら終了する。

【0042】図6と図7を用いて、指定した画面名称に対するサーブレットのソースコードを自動生成する処理の流れを説明する。サーブレットのソースコードを生成する際には、サーブレットのテンプレート701、パッケージ表702、対応表703、エラー対応表704、フォーム項目表706、およびBean公開メソッド表707を使用する。図7中、708は生成されるサーブレットのソースコードを示す。まず、対応表101から指定された画面名称102に対応する各名称を取得する(ステップ601)。ここで指定された画面名称102を“新規発注画面”として説明する。サーブレットファイル名称の“OrderServlet.java”をソースコード708のファイル名称としてファイルを作成する(ステップ602)。次に、テンプレートデータ208からサーブレットのテンプレート701を読み込む(ステップ603)。テンプレート701には、サーブレットの基本的な枠組みが記述されている。

【0043】次に、設計ドキュメント204からパッケージ表702を読み込み、package文709とimport文710を出力する(ステップ604)。package文709にはサーブレットのパッケージ名称を出力し、import文710にはBeanのパッケージ名称を出力する。パッケージ表702には各コンポーネントのパッケージ名称を記述してある。サーブレットファイル名称104“OrderServlet.java”から“OrderServlet”をクラス名称として取得し、クラス定義部分711を出力する(ステップ605)。“OrderServlet”が継承するスーパークラスのサーブレットとして、例えば、“BaseServlet”712を出力する。“BaseServlet”712は、本実施形態の開発支援システムで利用するクラスとして提供する。例えば、“names()”713と“business()”717は“BaseServlet”712で定義してあるメソッドであり、サブクラスで実装する。“names()”713はサーブレットで利用するJSP、Beanクラス、およびエラーJSPの名称を設定するメソッドで、“business()”717はサーブレットで処理する内容を定義するメソッドである。ステップ601で取得したJSP名称103“Order.jsp”を使用するJSP714として出力し

(ステップ606)、Beanファイル名称105“OrderBean.java”から使用するBeanクラス名称“OrderBean”715を取得し出力する(ステップ607)。そして、設計ドキュメント204からエラー対応表704を読み込み、“OrderServlet.java”に対するエラーJSP名称716を取得し出力する(ステップ608)。エラー対応表704は、各サーブレットファイル名称104とエラーJSP名称とを対応させた表で、ここでサーブレットを実行してエラーが発生した際にエラーを表示させるためのJSPを指定する。

【0044】次に、“business()”717の処理内容を出力する。ステップ607で取得した使用するBeanクラス名称を使って変数の設定を行う(ステップ609)。“bean”は

スーパークラス“BaseServlet”で定義されている変数で、サブクラスで使用するBeanクラスにキャストして利用する。そして、設計ドキュメント204から画面遷移図705を読み込み、画面名称から画面遷移元の画面を検索し(ステップ610)、設計ドキュメント204から遷移元の画面に対するフォーム項目表706を読み込む(ステップ611)。Webブラウザ302からサーバ304にリクエストを送信する際に、HTMLページ303のフォームに記述されている情報が入力データとして付加される。その入力データをフォーム項目表706で指定する。使用するBeanクラスのBean公開メソッド表707を設計ドキュメント204から読み込み(ステップ612)、フォームから渡される値を取得し、使用するBeanに値を設定するコード718を出力する(ステップ613)。Bean公開メソッド表707は使用するBeanクラスで公開している全メソッドの一覧で、フォーム項目表706に記述されている入力データに対するアクセスメソッドが定義されている。次に、フォーム項目表706に記述されている項目すべてに対して値を設定するコード718を出力したかをチェックする(ステップ614)。すべての項目に対して処理するまでステップ613の処理を行う。すべての項目を処理したら、“business()”717の定義を終了する。最後に、作成したソースコード708のファイルをサーブレットソースコードデータ209に格納し(ステップ615)、サーブレットのソースコードを自動生成する処理を終了する。

【0045】図7中、“BaseServlet”712はスーパークラスとして提供すると説明したが、図8により、スーパークラス“BaseServlet”712のソースコードを説明する。図8中、801はソースコードを示し、“BaseServlet”802がスーパークラスのクラス名称を示す。

【0046】まず、803はサーブレットで扱う変数の宣言で、JSP名称、Beanクラス名称、エラーJSP名称、および利用するBeanオブジェクトを宣言する。以下、804から811までメソッドの定義である。“init()”807と“doPost()”809は“HttpServlet”で定義されているメソッドで、このクラスで実装する。“business()”804はサブクラスでオーバーライドし、実際の処理を記述し、“attribute()”805はJSPからBeanをアクセスするための設定を行い、“forward()”806はサーブレットからJSPを呼び出すメソッドである。“doPost()”809で、804から806のメソッドの処理順序を決め、サブクラスでは処理順序を意識しないで済むようにしている。そのため、サブクラスを作成する開発者はサーブレットに関する特別な知識を必要としなくても済む。また、“init()”807からは、“names()”808を呼び出している。“names()”808は抽象メソッドとして定義してあり、サブクラスで実装する。図7中、714から716にあるようにサーブレットで使用するJSP、およびBeanクラスを設定する。“names()”808で、各名称を設定するだけで動作可能なサーブレットを作成することが可能になる。“error()”810は、サーブレットの

実行時にエラーが発生した場合の処理を定義したメソッドである。以降は、803で宣言した変数に対するアクセスメソッドの定義である。

【0047】図9と図10を用いて、指定した画面名称に対するJSPのソースコードを自動生成する処理の流れを説明する。JSPのソースコードを生成するに際しては、JSPのテンプレート1001、画面イメージ1002、対応表1003、画面遷移図1004、フォーム項目表1005、および入力処理表1006を使用する。図10中、1007は生成されるJSPのソースコードを示す。

【0048】まず、対応表101から指定された画面名称102に対応する各名称を取得する（ステップ901）。ここで指定された画面名称102を“新規発注画面”として説明する。JSP名称の“Order.jsp”をソースコード1007のファイル名称としてファイルを作成する（ステップ902）。次に、テンプレートデータ208からJSPのテンプレート1001を読み込む（ステップ903）。テンプレート1001には、JSPの基本的な枠組みが記述されている。

【0049】次に、ステップ901で取得したBeanファイル名称105“OrderBean.java”からBeanクラス名称1008“OrderBean”を取得し出力する（ステップ904）。〈jsp:useBean>1009は、JSP309内でBean 308を使用するためのJSP構文である。設計ドキュメント204から画面遷移図1004を読み込み、画面遷移先の画面を検索し（ステップ905）、遷移先の画面のサーブレットファイル名称104“OrderConfirmServlet.java”から、遷移先サーブレット1010に“OrderConfirmServlet”を設定する（ステップ906）。そして、設計ドキュメント204から入力処理表1005とフォーム項目表1006を読み込み（ステップ907）、画面イメージ1002のHTMLファイルを読み込む（ステップ908）。入力処理表1005はボタン押下やリスト選択などの際に処理する内容を入力チェック条件などと共に記述する。ステップ906で設定した遷移先サーブレット1010、および入力処理表1005とフォーム項目表1006の内容を、ステップ908で読み込んだHTMLファイルの内容に反映させて出力する（ステップ909）。最後に、作成したソースコード1007のファイルをJSPソースコードデータ210に格納し（ステップ910）、JSPのソースコードを自動生成する処理を終了する。

【0050】図11と図12を用いて、指定した画面名称に対するBeanのソースコードを自動生成する処理の流れを説明する。Beanのソースコードを生成するに際しては、Beanのテンプレート1201、パッケージ表1202、対応表1203、クラス図1204、シーケンス図1205、Bean公開メソッド表1206、およびフォーム項目表1207を使用する。図12中、1208は生成されるBeanのソースコードを示す。

【0051】まず、対応表101から指定された画面名称102に対応する各名称を取得する（ステップ1101）。ここで指定された画面名称102を“新規発注画面”として説明

する。Beanファイル名称の“OrderBean.java”をソースコード1208のファイル名称としてファイルを作成する（ステップ1102）。次に、テンプレートデータ208からBeanのテンプレート1201を読み込む（ステップ1103）。テンプレート1201には、Beanの基本的な枠組みが記述されている。

【0052】次に、設計ドキュメント204からパッケージ表1202を読み込み、package文1209を出力する（ステップ1104）。package文1209にはBeanのパッケージ名称を出力する。Beanファイル名称105“OrderBean.java”から“OrderBean”をクラス名称として取得し、クラス定義部分1210を出力する（ステップ1105）。設計ドキュメント204からクラス図1204を読み込み、“OrderBean”が継承するスーパークラスのBeanを検索し、そのクラス名称1211を出力する（ステップ1106）。そして、サーブレットから実行されるメソッドである“doBusiness()”1212の処理内容を定義する。設計ドキュメント204からシーケンス図1205を読み込み、シーケンス図1205に従って、“doBusiness()”1212を定義し出力する（ステップ1107）。クラス図1204とシーケンス図1205は、UML (Unified Modeling Language) 表記で記述されたドキュメントである。

【0053】次に、設計ドキュメント204からフォーム項目表1207とBean公開メソッド表1206を読み込み（ステップ1108）、フォーム項目表1207に記述されている項目すべてに対する変数宣言1213を出力し（ステップ1109）、Bean公開メソッド表1206のすべてのメソッドを定義し出力する（ステップ1110）。最後に、作成したソースコード1208のファイルをBeanソースコードデータ211に格納し（ステップ1112）、Beanのソースコードを自動生成する処理を終了する。

【0054】図13を用いて、Beanは画面に表示するために必要な情報をすべて持っているコンポーネントであることを説明する。図13は、画面情報とBeanとの関係を表している。図13中、1301は画面イメージであり、“新規発注確認画面”を例にして説明する。1302はBeanのクラスを示す。クラス1302の上の区画1303に表示している名称はクラス名で、“新規発注確認画面”に対応する“OrderConfirmBean”がBeanのクラス名であることを示す。下の区画1304はクラスの属性を表す。“OrderConfirmBean”は“新規発注確認画面”を表示するために必要となる情報をすべて保持し、それらの情報はJSPから参照され、HTMLページとして表示される。画面に表示するために必要な情報、およびフォームで入力される情報は、図12中のフォーム項目表1207とBean公開メソッド1206とで定義されている。

【0055】次に、本発明の第2の実施の形態を説明する。ここでは、図14で後述するサーブレット、JSP、およびBeanを利用したWebアプリケーションシステムを設計開発する例で説明する。特に、本発明は、設計情報にある画面仕様の各画面に対応させて、サーブレット、

JSP、およびBeanの各コンポーネントの名称を対応させて定義することを特徴としている。この対応の定義と、設計情報の内容を利用して、メソッドの定義、および各コンポーネント間の呼び出し関係を自動的に抽出し、ソースコードを自動生成する。

【0056】図14は、本実施の形態において設計開発したサーブレット、JSP、およびBeanを利用したWebアプリケーションシステムの全体構成の一例を示した図である。このWebアプリケーションシステムは、クライアント1401からWebブラウザ1402によって利用される。Webブラウザ1402上には、HTMLページ1403が表示される。Webブラウザ1402からのリクエストはHTTP経由でサーバ側1404に送信される。サーバ1404には、HTTPサーバ1405とアプリケーションサーバ1406が設けられており、アプリケーションサーバ1406には、サーブレットやJSPの実行環境が含まれている。

【0057】サーブレット1407、JSP 1408、およびBean 1409は、サーバ1404側のアプリケーションサーバ1406上に配置され、クライアント1401側のWebブラウザ1402からリクエストが送信されると、リクエストに応じて該当するサーブレット1407が起動される。サーブレット1407は、対応するBean 1409に処理を要求する。さらに、Bean 1409は、業務クラスおよびDBアクセスクラス1410を利用して、データベース1411にアクセスし、業務処理に必要なデータ操作を行い、業務に関する処理やデータ加工などを行う。Bean1409は、その処理結果を保持しておく。業務クラス、DBアクセスクラス1410は、業務を分析、設計した結果、作成されるクラスや、再利用可能な既存のクラスである。次にサーブレット1407は、JSP 1408にBean1409を渡し、JSP 1408を呼び出し、HTMLページの出力を依頼する。JSP 1408は、Bean 1409で保持している処理結果を参照して、HTMLページを生成し、クライアント1401側に返信する。返信されたHTMLページ1403は、Webブラウザ1402上に表示される。

【0058】前述したようにサーブレット1407は、Webブラウザ1402からのリクエストの受け付け、Bean 1409への処理要求、およびJSP 1408の呼び出しという、それぞれの間をつなぎ、制御する役割を持つ。JSP 1408はHTMLページを出力するための表示に関することを受け持ち、Bean 1409は業務クラス、DBアクセスクラス1410を利用して業務を組み立て、実行する処理と、その処理結果の保持ということを受け持つ。それぞれの役割を明確に分離することで、機能分担を簡潔に把握することができる。

【0059】図15は、本実施形態のWebアプリケーションシステムにおける受信コンポーネントと応答コンポーネントの関係をいくつか示している。入力画面1510は、サーバ1404にリクエストを送信する画面を示す。このリクエストがサーバ側で処理され、その結果が出力画面1504として、クライアント1401に返信される。図14

中、サーブレット1407が受信コンポーネント1502に相当し、JSP 1408が応答コンポーネント1503に相当する。図14は、サーブレット、JSP、およびBeanを利用したWebアプリケーションシステムの構成であるが、JSP 1408やBean 1409を利用せず、サーブレット1407だけ、またはJSP 1408だけでWebブラウザ1402からのリクエストを受け、処理を返すことも可能である。受信窓口部品1(1505)や応答窓口部品1(1506)を用意することで、受信したリクエストに応じて、対応する受信部品に振り分けたり、処理結果に応じて、対応する応答部品に振り分けることもできる。従って、受信コンポーネントと応答コンポーネントが1対1とは限らない。開発対象のWebアプリケーションシステムの構成に従って、適切な関係を選択して、コンポーネントを定義すればよい。ここでは、受信部品7(1507)と応答部品7(1508)のように、受信コンポーネントと応答コンポーネントが1対1である場合について説明する。

【0060】図16と図17は、受信コンポーネント1502と応答コンポーネント1503の対応を1対1に定義した例を示す。イベント1601は、入力画面1501からサーバにリクエストが送信される際の、ボタン押下などのイベントを示す。条件1602は、サーバ側で処理した結果、受信コンポーネント1502が応答コンポーネント1503を呼び出す際の条件を示す。条件1602が設定されている場合は、例外として、受信コンポーネント1502と応答コンポーネント1503との対応は1対nとなる。

【0061】図16では、入力画面1501からのリクエストを受ける受信コンポーネント1502は、出力画面1504に依存している。入力画面1501からのリクエストは、次の出力画面1504を出力するものとして、受信コンポーネント1502と応答コンポーネント1503は、出力画面1504に対応して作成される。入力画面1501の画面C 1603では、イベント1601によって、リクエストを送信する受信コンポーネント1502が異なる。

【0062】一方、図17では、受信コンポーネント1502は入力画面1501に依存している。入力画面1501からのリクエストは、対応する受信コンポーネント1502が受ける。その受信コンポーネント1502が、イベント1601や条件1602によって、応答コンポーネント1503を振り分ける。受信コンポーネント1502と応答コンポーネント1503は、それぞれ、入力画面1501、および出力画面1504に対応して作成される。入力画面1501の画面C 1701のように、イベント1601が複数ある場合でも、対応する受信コンポーネント1502が受けて処理する。以下では、図16で示したコンポーネント対応の定義をした場合について説明する。

【0063】図18は、本実施の形態において設計開発するWebアプリケーションシステムの画面名称と、サーブレット、JSP、およびBeanとの対応関係の定義を表形式に表した一例を示す図である。対応定義表1801は、画

面名称1802と画面ID1803、および、サーブレット名称1804とBean名称1805とJSP名称1806とから成り、それぞれ画面に対応して名称を定義する。画面名称1802は、サーブレット名称1804とBean名称1805とJSP名称1806の組み合わせからなるコンポーネントによって出力される画面の名称を示す。サーブレット名称1804で与えられるサーブレットが受信コンポーネント1502であり、JSP名称1806で与えられるJSPが応答コンポーネント1503となる。Bean名称1805で与えられるBeanは、サーブレットから要求された処理を実行するコンポーネントであり、JSPがHTMLページを生成する際に情報を参照するコンポーネントである。

【0064】サーブレット名称1804とBean名称1805は、拡張子を付けず、クラスファイルの名称とし、JSP名称1806の拡張子は、".jsp"とする。例えば、画面名称1802が"トップ"1807に対して、サーブレット名称1804は、画面ID1803"default"の先頭一文字を大文字にし"Servlet"を付加した"DefaultServlet"とし、同様に、Bean名称1805は、先頭を大文字にした画面ID1803に"Bean"を付加した"DefaultBean"とする。JSP名称1806は、画面ID1803に拡張子".jsp"を付けた"default.jsp"とする。

【0065】サーブレット名称1804とBean名称1805との関係、あるいはJSP名称1806とBean名称1805との関係が、"発注確認"1809のように、n対mになる場合がある。この場合、Bean名称1805は、画面ID1803などからは定義できないため、編集して追加することになる。また、"エラー表示"1810の場合、JSP名称1806の"error.jsp"1811だけを定義している。これは、いくつかのサーブレットから呼び出されるものである。以下では、サーブレット名称1804、Bean名称1805、およびJSP名称506の関係が1対1の場合について説明する。

【0066】図19に示す対応定義表1901は、それぞれの画面名称1802に対応して、1対1の関係で、サーブレット名称504、Bean名称505、およびJSP名称506を定義した例を示す。

【0067】図20は、Webアプリケーションシステムを設計開発するシステムのシステム構成の一例を示した図である。図20中、2001はコマンド入力やマウスを使った操作を行う端末で、2002は後述する各種の機能を実現する中央処理装置で、2003は様々な情報を格納する外部記憶装置を示す。外部記憶装置2003に格納されている設計情報2004は、Webアプリケーションシステムの開発において、顧客からの要求や業務仕様から得られる画面仕様や設計書などの設計ドキュメント、および画面デザインや各画面イメージを表すHTMLファイルなどを格納したものである。対応定義データ2005には、図19で示したコンポーネントの対応定義表1901が格納されている。テンプレートデータ2006には、各コンポーネントのソースコードを生成する際のテンプレートや、サーブレットやBeanのスーパークラスとなるソースコード、クラスフ

ファイルが格納されている。ソースコードデータ2007には、サーブレット、JSP、およびBeanのソースコードが格納される。

【0068】中央処理装置2002には、次の3つの機能を実現するソフトウェアが格納され動作している。対応定義生成/編集部2008は、設計情報2004に格納されている画面仕様から画面情報を読み込み、画面とサーブレット、JSP、およびBeanのコンポーネントとの対応定義表1901を生成し、対応定義データ2005に格納する。さらに、対応定義生成/編集部2008を利用して、端末2010から、生成された対応定義表1901の編集を行うことができる。コード生成部2009は、対応定義データ2005に格納されている対応定義表1901と、画面仕様やシステム設計書などの設計情報2004を利用して、サーブレット、JSP、およびBeanの各ソースコードを自動生成する。ソースコードを生成する際に、テンプレートデータ2006に格納されている各コンポーネントのテンプレートやスーパークラスを利用する。コード生成部2009によって生成される各ソースコードは、ソースコードデータ2007に格納される。コード編集部2010は、対応定義データ2005に格納されている対応定義表1901を利用して、開発対象の画面に対する各コンポーネントのソースコードをソースコードデータ2007から取得し、端末2001からエディタを利用して編集する。

【0069】図21は、本実施形態のWebアプリケーションシステムを設計開発するシステムでの開発作業手順の一例を示した図である。まず、サーブレットの作成を行い、順に、Bean、JSPと作成を進める。サーブレットの作成においては、サーブレットのテンプレート2101を元にして、設計情報2102を利用して、サーブレットのソースコード2108を作成する。サーブレットを作成するために必要な設計情報2102には、例えば、対応定義表1901、パッケージ仕様2104、画面遷移図2105、送信データ仕様2106、セッション管理仕様2107、およびエラー処理仕様2108がある。

【0070】パッケージ仕様2104には、各コンポーネントのパッケージ定義とその名称が記述してある。画面遷移図2105は画面間の遷移を示した図である。Webブラウザからサーバにリクエストを送信する際に、HTMLページのフォームに記述された内容が送信データに付加される。その送信データの内容を送信データ仕様2106に記述する。セッション管理仕様2107には、画面間で引き継ぐ情報を記述する。エラー処理仕様2108には、サーバ側で処理した結果、エラーが発生した場合に行う処理の内容を記述する。

【0071】次に、Beanの作成を行う。Beanの作成においては、Beanのテンプレート2109を元にして、設計情報2110を利用して、Beanのソースコード2111を作成する。Beanを作成するために必要な設計情報2110には、例えば、対応定義表1901、パッケージ仕様2104、画面遷移図

2105、送信データ仕様2106、チェック項目2112、クラス仕様2113、メソッド仕様2114、およびシーケンス図2115がある。

【0072】チェック項目2112には、リクエストが送信された際の送信データの値チェックや妥当性チェックなどの項目を記述する。クラス仕様2113は、Beanや図14中の業務クラス、DBアクセスクラス1410のクラス図やクラスの概要、属性、メソッドを記述したものである。メソッド仕様2114には、クラス仕様2113のクラスが持つメソッドの詳細を記述する。シーケンス図2115は、メソッドの処理の流れを図で示したものである。クラス仕様2113のクラス図とシーケンス図2115はUML (Unified Modeling Language) 表記で記述されたドキュメントである。

【0073】最後に、JSPの作成を行う。JSPの作成においては、JSPのテンプレート2116を元にして、設計情報2117を利用して、JSPのソースコード2118を作成する。JSPを作成するために必要な設計情報2117には、例えば、対応定義表1901、画面イメージ2119、画面遷移図2105、入力処理仕様2120、送信データ仕様2106、クラス仕様2113、およびメソッド仕様2114がある。

【0074】画面イメージ2119は、画面仕様の各画面のイメージを定義したもので、HTMLファイルで作成する。入力処理仕様2120には、HTMLページのフォームでの、ボタン押下やリスト選択などでの入力チェックであるクライアント側でのチェック項目や処理を記述する。

【0075】本実施形態のシステムにおいては、コード生成部2009によって、それぞれのサブリット、Bean、およびJSPのソースコード2103、2111、2118が自動生成され、外部記憶装置2003のソースコードデータ2007として格納される。また、コード編集部2010によって、ソースコードの編集を行う。

【0076】図22を参照して、対応定義部/編集部2008によって、対応定義表1901を生成する処理の流れを説明する。まず、設計情報2004に格納されている画面仕様から全ての画面情報を読み込む(ステップ2201)。最初の画面情報から画面名称と画面IDを取得する(ステップ2202)。画面IDは、例えば、各画面イメージとして作成されているHTMLファイルのファイル名称から取得してもいいし、画面情報の中で定義してあってもよい。また、端末2001から画面毎に入力してもよい。ここでは、取得した画面名称を“発注確認”、画面IDを“confirm”として説明する。

【0077】次に、取得した画面IDの先頭一文字を大文字にして、“Servlet”を付加し、サブリット名称とする(ステップ2203)。同様に、画面IDの先頭一文字を大文字にして、“Bean”を付加し、Bean名称とする(ステップ2204)。画面IDに拡張子“.jsp”を付けてJSPのファイル名称とする(ステップ2205)。ここで、各コンポーネントの名称は、それぞれ、“ConfirmServlet”、“ConfirmBean”、“confirm.jsp”と定義される。画面名称と画

面ID、定義されたサブリット名称、Bean名称、およびJSP名称の各名称の一組を対応定義表1901に追加する(ステップ2206)。

【0078】次に、ステップ2201で取得した画面情報の全てに対して処理したかをチェックし(ステップ2207)、全て処理するまで、ステップ2202からステップ2206までの処理を繰り返す。全ての画面情報に対して処理を行ったら、最後に、対応定義表1901を対応定義データ2005に格納して終了する(ステップ2208)。ここで生成された対応定義表1901は、設計情報2004に格納してもよい。

【0079】次に、図23を参照して、サブリット、JSP、およびBeanの各コンポーネントのソースコードをコード生成部2009によって自動生成する処理の流れを説明する。まず、対応定義生成/編集部2008によって生成した対応定義表1901を読み込む(ステップ2301)。対応定義表1901から画面名称1802を取得する(ステップ2302)。ここで取得した画面名称を“発注確認”として説明する。画面名称に対応する各コンポーネントの名称を取得する(ステップ2303)。取得した画面名称に対応するサブリット名称1804、Bean名称1805、およびJSP名称1806の各名称が全て設定されているかチェックする(ステップ2304)。例えば、図19中の“エラー表示”1811の場合、JSP名称1806“error.jsp”のみであるため、ステップ2304からステップ2302に戻る。各名称が全て設定している場合、以降の処理を行う。

【0080】まずステップ2302で取得した画面名称である“発注確認”に対するサブリットのソースコード“ConfirmServlet.java”を生成する(ステップ2305)。同様に、Beanのソースコード“ConfirmBean.java”を生成し(ステップ2306)、JSPのソースコード“confirm.jsp”を生成する(ステップ2307)。ステップ2301で読み込んだ対応定義表1901の全ての画面名称に対して処理したかをチェックし(ステップ2308)、全て処理するまで、ステップ2302からステップ2307までの処理を繰り返す。全ての画面名称に対して処理を行ったら終了する。

【0081】なお、ここではステップ2304で画面名称に対応するサブリット名称1804、Bean名称1805、およびJSP名称1806の各名称が全て設定されていることをチェックした後、ソースコードの生成を行っているが、ステップ2304のチェックは必須ではない。サブリット名称1804、Bean名称1805、およびJSP名称1806の何れかの名称が設定されていれば、それについてソースコードの生成を行うようにしてもよい。

【0082】図24と図25を用いて、指定した画面名称に対するサブリットのソースコードをコード生成部2009で自動生成する処理の流れを説明する。対応定義表1901で定義されている各コンポーネントの名称は予め与えられているとする。ここで指定された画面名称1802を“発注確認”、各コンポーネントの名称を“ConfirmServ

let”、“ConfirmBean”、“confirm.jsp”として説明する。サブリットのスソースコードを生成するに際しては、サブリットのスソースコードのテンプレート2101と、設計情報2004に格納されている設計ドキュメントから対応定義表1901、パッケージ仕様2104、画面遷移図2105、送信データ仕様2106、セッション管理仕様2107、およびエラー処理仕様2108などを使用する。図25中、2501は生成されるサブリットのスソースコードを示す。

【0083】まず、与えられたサブリット名称1804をファイル名称として、拡張子“.java”でファイル“ConfirmServlet.java”を作成する(ステップ2401)。次に、テンプレートデータ2006からサブリットのスソースコードの一覧を読み込む(ステップ2402)。テンプレートデータ2006には、サブリットのスソースコードの基本的な枠組みの他、いろいろな用途に応じたサブリットのスソースコードのテンプレートや、スーパークラスが格納されている。読み込んだテンプレート一覧から、作成するサブリットに該当するテンプレート2101を選択する(ステップ2403)。選択したテンプレート2101を雛型として、スソースコード2501を生成する。ここで、使用するテンプレート2101は、個々に選択する方法の他、既に設定されている場合もある。

【0084】次に、パッケージ仕様2104を読み込み、package文2502とimport文2503を出力する(ステップ2404)。package文2502には、サブリットのスソースコードのパッケージ名称を出力し、import文2503には、Beanのパッケージ名称を出力する。そして、サブリット名称1804“ConfirmServlet”をクラス名称として、クラス定義2504に出力する(ステップ2405)。“service()”2505はサブリットAPIが提供するメソッドで、クライアントから要求があった場合、このメソッドが実行される。以降の処理では、“service()”2505の処理内容を出力する。

【0085】まず、与えられたBean名称1805“ConfirmBean”をサブリットから生成し、使用するBeanクラス名称2506として出力する(ステップ2406)。画面遷移図2105から遷移元の画面を検索し(ステップ2407)、遷移元の画面の送信データ仕様2106を読み込み、パラメータを取得する際の項目名2507とし、その取得とBeanへの設定を出力する(ステップ2408)。次に、セッション管理仕様2107を読み込み(ステップ2409)、セッションから取得する値の項目名2508を出力する(ステップ2410)。同様に、セッションに格納する値と、項目名2510を出力する(ステップ2411)。“doTask()”2509は、サブリットからBeanに処理を要求するためのメソッドである。そして、エラー処理仕様2108を読み込み、エラー時の処理内容2511を出力する。“service()”2505の最後に、与えられたJSP名称1806“confirm.jsp”をサブリットから呼び出すJSP名称2512として出力する(ステップ2413)。最後に、生成したスソースコード2501のファイルをスソースコードデータ2007に格納し、サブリットのスソースコードを自動生成する処理を終了する(ステップ2414)。

【0086】図24と図25では、サブリットのスソースコードの生成例として、テンプレートデータ2101を雛型として、設計情報2004の内容から、必要な項目を自動的に抽出し、適切な箇所に出力することで生成した。次に図26を用いて、スソースコードを生成する際のテンプレートに、本実施形態の開発支援システムで利用するクラスとしてスーパークラスを用意しておき、それを継承して作成することもできることを説明する。図26中、2601は生成されるサブリットのスソースコードを示す。多くの共通な処理はスーパークラスで定義されているため、未実装または、差分だけを開発すればよい。また、図25のスソースコード2501と比べ、図26のスソースコード2601は簡単に生成することができる。

【0087】スーパークラスを継承して作成する場合、図24中、ステップ2403で、利用するテンプレート2101にスーパークラスを選択し、それをステップ2405でクラス定義を出力する際に、継承するスーパークラス名称2602に出力する。サブリットのスソースコードのスーパークラスでは、ほとんどの処理が実装済みで、例えば、“init()”2603、“beforeTask()”2604、“afterTask()”2605は宣言のみしてあるものである。これらのメソッド2603、2604、2605は、サブクラスで定義する。“init()”2603は、初期化時の設定を行うメソッドで、“setBeanName()”2606や“setJspPage()”2607を利用して、サブリットから使用、または、呼び出すBean名称やJSP名称を指定する。そして、“beforeTask()”2604は、サブリットからBeanに処理を要求する前に行う処理内容を実装し、“afterTask()”2605には、Beanの処理が終了した後に行う処理内容を実装する。例えば、セッション管理などをここで行う。このように、開発者は、“init()”2603、“beforeTask()”2604、“afterTask()”2605のみを実装すればサブリットを開発できる。

【0088】図26中、テンプレートに、本実施形態の開発支援システムで利用するクラスとしてスーパークラスを用意しておき、それを継承して生成すると説明したが、図27を用いて、スーパークラスのスソースコード例を説明する。図27中、2701はスーパークラスのスソースコードを示し、クラス名称2702は“BaseServlet”である。

【0089】まず、2703は“BaseServlet”で扱う変数の宣言で、JSP名称とBean名称、および、使用するBeanオブジェクトを宣言する。以下、2704から2712までメソッドの定義である。“init()”2704は、初期化時の設定を行うメソッドで、サブクラスで実装し、サブリットから使用、または、呼び出すBean名称やJSP名称を設定する。“service()”2705は、サブリットAPIが提供するメソッドで、クライアントから要求があった場合、このメソッドが実行される。“service()”2705では、2707から2712までのメソッドを決められた順序で実行する。そのため、このスーパークラス“BaseServlet”2702を継承し

たサブクラスでは、処理順序を意識しない済む。そのため、サブクラスを開発する開発者はサーブレットに関する特別な知識を必要としない。サブクラスで、“init()”2704を定義すれば、動作可能なサーブレットを作成することができる。

【0090】“create()”2706は、指定されたBean名称のオブジェクトを生成し、“doTask()”2708はBeanに処理要求を行う。“doTask()”2708の前後での処理を“beforeTask()”2707と“afterTask()”2709に記述し、必要であれば、サブクラスで実装する。“callJsp()”2710は、指定されたJSP名称のJSPを呼び出す。“service()”2705の実行中に例外が発生した場合は、“error()”2711が実行され、エラー処理を行う。2712は、“BaseServlet”で扱う変数2703に対するアクセスメソッドの定義である。

【0091】サーブレットのソースコードをコード生成部2009で自動生成する際、図24中、ステップ2403で、作成するサーブレットに該当するテンプレートを選択するが、図28を用いて、テンプレート選択方法の一例について説明する。図28中、2801は、本実施形態の開発支援システムの操作画面である。画面の左側の領域2802

には、設計開発の対象とするシステム名称“発注システム”2803が表示されている。その下に階層構造で、画面名称の“発注確認”2804が表示され、さらにサーブレット、Bean、およびJSPが表示される。この階層構造は、対応定義生成/編集部2008によって定義された対応定義表1901の内容を表現している。

【0092】ここで、サーブレットの“ConfirmServlet”2802を生成する際に使用するテンプレートを選択する場合で説明する。2806はサーブレットのテンプレートを選択させるテンプレート選択画面であり、テンプレートデータ2006から読み込まれたテンプレート一覧がリスト表示されている。リスト表示には、テンプレートを示す名称2807とコード名称2808が表示されている。テンプレート選択画面2806上で、テンプレートを選択し、実行ボタン2809を実行すると、選択されているテンプレートが使用される。また、内容説明ボタン2810を実行すると、選択されているテンプレートに関する詳細な説明を示した画面2811が表示される。

【0093】図30と図33で後述するBeamとJSPのソースコードを生成する際にも、サーブレットと同様に、テンプレートを選択し、それを雛型として利用する。図29を用いて、BeanとJSP、それぞれのテンプレート選択方法の一例について説明する。図28中、テンプレート選択画面2806と同じように、Beanのテンプレート選択画面2901、JSPのテンプレート選択画面2904がある。それぞれの画面のリスト表示には、テンプレートを示す名称2902、2905とコード名称2903、2906が表示されている。Beanのテンプレート選択画面2901ではクラスの構成がクラス階層で示されている。また、JSPのテンプレート選択画面2904では、テンプレートを複数選択すること

もでき、テンプレートを組み合わせて利用することができる。

【0094】図30と図31を用いて、指定した画面名称に対するBeanのソースコードをコード生成部2009で自動生成する処理の流れを説明する。対応定義表1901で定義されている各コンポーネントの名称は予め与えられているとする。以下では、指定された画面名称1802を“発注確認”、Beanのコンポーネントの名称を“ConfirmBean”として説明する。Beanのソースコードを生成するに際しては、JSPのテンプレート2109と対応定義表1901と、設計情報2004に格納されている設計ドキュメントからパッケージ仕様2104、画面遷移図2105、送信データ仕様2106、チェック項目2112、クラス仕様2113、メソッド仕様2114、およびシーケンス図2115などを使用する。図31中、3101は生成されるBeanのソースコードを示す。

【0095】まず、与えられたBean名称1805をファイル名称として、拡張子“.java”でファイル“ConfirmBean.java”を作成する（ステップ3001）。次に、テンプレートデータ2006からBeanのテンプレート一覧を読み込む（ステップ3002）。テンプレートデータ2006には、サーブレットのテンプレートと同様に、Beanの基本的な枠組みの他、いろいろな用途に応じたBeanのテンプレートや、スーパークラスが格納されている。読み込んだテンプレート一覧から、作成するBeanに該当するテンプレート2109を選択する（ステップ3003）。選択したテンプレート2109を雛型として、ソースコード3101を生成する。

【0096】次に、パッケージ仕様2104を読み込み、package文3102を出力する（ステップ3004）。package文3102には、Beanのパッケージ名称を出力する。そして、Bean名称1805“ConfirmBean”をクラス名称として、クラス定義3103に出力する（ステップ3005）。さらに、クラス仕様2113から対象のBeanクラス“ConfirmBean”のスーパークラスを検索し、その名称を出力する（ステップ3006）。

【0097】“doTask()”3104はサーブレットから実行されるメソッドで、ここにBeanで処理内容3105を実装する。シーケンス図2115を読み込み、そのシーケンス図2115に従って、処理内容を出力する（ステップ3007）。そして、“check()”3106の処理内容を出力する。“check()”3106は、“doTask()”3104から呼び出されるメソッドで、サーブレットから渡されるパラメータのチェックを行う。まず、画面遷移図2105から遷移元の画面を検索し（ステップ3008）、遷移元の画面の送信データ仕様2106を読み込み、パラメータを取得する際の項目名3107を出力する（ステップ3009）。チェック項目2112を読み込み、ステップ3009で出力したパラメータについてのチェック処理3108を出力する（ステップ3010）。

【0098】次に、クラス仕様2113から、対象のBeanクラス“ConfirmBean”の属性を取得し、その属性の宣言3109を出力し（ステップ3011）、その属性に対する取得メ

ソッドと設定メソッド3110を出力する（ステップ3012）。さらに、メソッド仕様2114を読み込み、“doTask()”3104と“check()”3106と属性の設定メソッドと取得メソッド3110以外のメソッドを出力する（ステップ3013）。最後に、生成したソースコード3101のファイルをソースコードデータ2007に格納し、Beanのソースコードを自動生成する処理を終了する（ステップ3014）。

【0099】図32を用いて、Beanは画面に表示するために必要な情報を全て保持しているコンポーネントであることを説明する。図32は、画面情報とBeanとの関係を表している。図32中、3201は画面イメージであり、ここでは“発注確認”画面を例にして説明する。3202はBeanのクラスを示す。クラス3202の上の区画3203に表示している名称はクラス名で、“発注確認”に対応する“ConfirmBean”がBeanのクラス名であることを示す。下の区画3204はクラスの属性を表す。“ConfirmBean”には、“発注確認”画面を表示するために必要となる情報が属性として全て保持され、それらの属性はJSPから参照され、HTMLページとして表示される。画面に表示するために必要な情報、および、HTMLページ上のフォームで入力される送信される情報は、設計情報2004に格納されている設計ドキュメントの送信データ仕様2106に記述されている。

【0100】図33と図34を用いて、指定した画面名称に対するJSPのソースコードをコード生成部2009で自動生成する処理の流れを説明する。対応定義表1901で定義されている各コンポーネントの名称は予め与えられているとする。ここでは、指定された画面名称1802を“発注確認”、BeanとJSPのコンポーネントの名称を“ConfirmBean”、“confirm.jsp”として説明する。JSPのソースコードを生成するに際しては、JSPのテンプレートと対応定義表1901と、設計情報2004に格納されている設計ドキュメントから画面イメージ2119、画面遷移図2105、入力処理仕様2120、送信データ仕様2106、クラス仕様2113、およびメソッド仕様2114などを使用する。図34中、3401は生成されるJSPのソースコードを示す。

【0101】まず、与えられたJSP名称1806をファイル名称として、ファイル“confirm.jsp”を作成する（ステップ3301）。設計ドキュメントから“発注確認”画面の画面イメージ2119のHTMLファイルを読み込み（ステップ3302）、読み込んだHTMLファイルの内容を出力する（ステップ3303）。次に、テンプレートデータ2006からJSPのテンプレート一覧を読み込む（ステップ3304）。テンプレートデータ2006には、JSPの基本的な枠組みの他、いろいろな用途に応じたJSPのテンプレートが格納されている。読み込んだテンプレート一覧から、作成するJSPに該当するテンプレート2116を選択する（ステップ3305）。選択したテンプレート2116を雛型として、ステップ3303で出力したHTMLの内容を加工してソースコード3401を生成する。

【0102】次に、与えられたBean名称1805“ConfirmBe

an”をJSPで利用するクラス3402として出力する（ステップ3306）。<jsp:useBean>タグ3402は、JSP内でBeanを使用するための構文である。入力処理仕様2120を読み込み、クライアント側でのチェック項目に対するスクリプト定義3403を出力する。そして、画面遷移図2105から遷移元の画面を検索し（ステップ3308）、遷移元の画面に対するサーブレット名称1804を対応定義表1901から検索し、フォームタグのアクション指定3404に出力する（ステップ3309）。次に、送信データ仕様2106を読み込み、フォーム内のテキスト入力やボタンなどの部品の名前3405を出力する（ステップ3310）。そして、クラス仕様2113を読み込み、フォーム内の部品の値やテキストにBeanの属性名をJSPの構文を使用して、該当箇所へ出力する（ステップ3311）。ステップ3311と同様に、メソッド仕様2114を読み込み、Beanのメソッドを使用して、該当箇所へ出力する（ステップ3312）。<jsp:getProperty>タグ3406や<%= %>タグ3407は、<jsp:useBean>タグ3402と同様、JSPの構文である。最後に、生成したソースコード3401のファイルをソースコードデータ2007に格納し、JSPのソースコードを自動生成する処理を終了する（ステップ3313）。

【0103】図35を参照して、サーブレット、JSP、およびBeanの各コンポーネントのソースコードをコード編集部2010によって編集する方法の一例について説明する。図35中、2801は、本実施形態の開発支援システムの操作画面である。画面の左側の領域2802には、階層構造で、対応定義表1901の内容が表示されている。ここで、“発注確認”2804を選択し、例えば、ダブルクリックすると、“発注確認”2804に対する各コンポーネントのソースコードが画面の右側の領域3501にエディタが開き表示される。サーブレットのソースコード“ConfirmServlet.java”はエディタ画面3504に開かれ、同様に、Beanのソースコード“ConfirmBean.java”はエディタ画面3503に、JSPのソースコード“confirm.jsp”はエディタ画面3504に開かれる。それぞれのソースコードを開く際には、対応定義表1901を利用して、編集対象となる各コンポーネントのソースコードをソースコードデータ2007から読み込む。また、“ConfirmServlet”2805を単独で選択して、“ConfirmServlet.java”を開くこともできる。これによって、開発者は、開発対象の画面に関係するソースコードを、画面名称を選択して瞬時に開き、参照、または、編集することができる。

【0104】

【発明の効果】以上、説明したように、本発明によれば、以下の効果が得られる。

(1) 設計ドキュメントを記述し、各コンポーネントの名称の対応表を作成するだけで、システム開発を容易に行うことができ、かつ、開発作業を軽減できるため、開発期間を短縮することができる。

(2) 各画面に対するサーブレット、JSP、およびBean

の各コンポーネントの対応が明確で、それぞれの役割が分離できているため、画面単位に分担開発が行える。

(3) 各コンポーネントのソースコードを自動生成することによって、Javaの特別な知識を必要とせず、プログラミングを簡単にすることができ、開発作業を軽減することができる。自動生成するソースコードに関して、コード自動生成部によって、標準化した統一された形でコードを生成できるためメンテナンスは容易になり、性能面でも劣化することなく保証されたプログラムを提供することができる。

(4) システムのバージョンアップにおける機能追加や拡張、修正でも、サーブレット、JSP、およびBeanの各コンポーネントを新たに定義して追加する作業であったり、修正対象、修正範囲にあるコンポーネントの特定が容易であるため、拡張性のあるシステムを構築できる。また、この本発明の開発支援システムが提供するものは、サーブレット、JSP、およびBeanで構成するWebアプリケーションシステムのフレームワークであるともいえる。

(5) Webアプリケーションシステムの開発に際して、画面仕様が決定することで、各コンポーネントを定義できるため、プロトタイプ作成や、早い時期に実際に動作するプログラムでアーキテクチャを検証することができる。

【図面の簡単な説明】

【図1】開発するWebアプリケーションの画面とサーブレット、JSP、およびBeanとの対応表の一例を示す図

【図2】本実施の形態のシステム構成の一例を示す図

【図3】サーブレット、JSP、およびBeanを利用したWebアプリケーションシステムの全体構成の一例を示す図

【図4】対応表の生成処理の流れを示すフローチャートの一例を示す図

【図5】ソースコードの自動生成処理の流れを示すフローチャートの一例を示す図

【図6】サーブレットのソースコードを自動生成する処理の流れを示すフローチャートの一例を示す図

【図7】サーブレットのソースコード生成を説明するための図

【図8】サーブレットのスーパークラスのソースコードの一例を示す図

【図9】JSPのソースコードを自動生成する処理の流れを示すフローチャートの一例を示す図

【図10】JSPのソースコード生成を説明するための図

【図11】Beanのソースコードを自動生成する処理の流れを示すフローチャートの一例を示す図

【図12】Beanのソースコード生成を説明するための図

【図13】画面情報とBeanの関係を示す図

【図14】サーブレット、JSP、およびBeanを利用したWebアプリケーションシステムの全体構成の第2の実施形態を示した図

【図15】第2の実施形態のWebアプリケーションシステムにおける受信コンポーネントと応答コンポーネントの関係を示す図

【図16】受信コンポーネントと応答コンポーネントの対応を1対1に定義した例(その1)を示す図

10 【図17】受信コンポーネントと応答コンポーネントの対応を1対1に定義した例(その2)を示す図

【図18】コンポーネントの対応定義表の例(その1)を示す図

【図19】コンポーネントの対応定義表の例(その2)を示す図

【図20】Webアプリケーションシステムを設計開発するシステムのシステム構成の一例を示した図

【図21】Webアプリケーションシステムを設計開発するシステムでの開発作業手順の一例を示した図

20 【図22】対応定義表を生成する処理の流れを示すフローチャート図

【図23】ソースコードの自動生成処理の流れを示すフローチャート図

【図24】サーブレットのソースコードを自動生成する処理の流れを示すフローチャート図

【図25】サーブレットのソースコード生成例(その1)を示す図

【図26】サーブレットのソースコード生成例(その2)を示す図

30 【図27】サーブレットのスーパークラスのコード生成例を示す図

【図28】テンプレートの選択イメージ(その1)を示す図

【図29】テンプレートの選択イメージ(その2)を示す図

【図30】Beanのソースコードを自動生成する処理の流れを示すフローチャート図

【図31】Beanのソースコード生成例を示す図

【図32】画面情報とBeanの属性の関係を示す図

40 【図33】JSPのソースコードを自動生成する処理の流れを示すフローチャート図

【図34】JSPのソースコード生成例を示す図

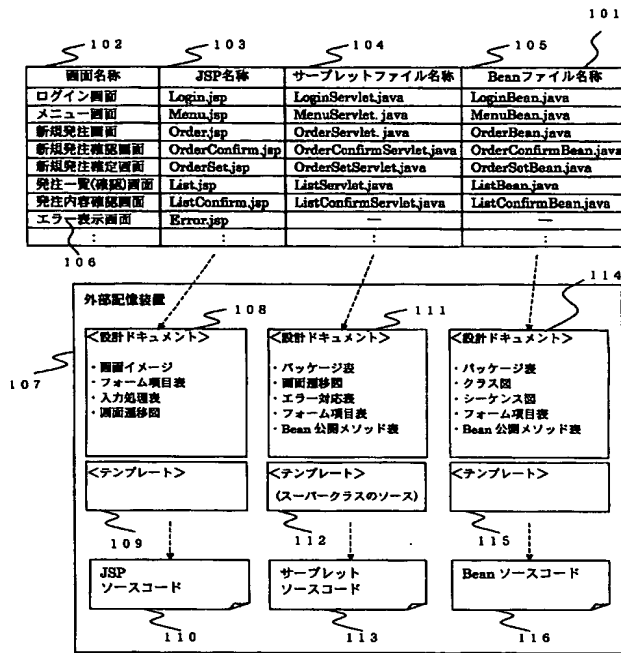
【図35】ソースコードの編集イメージを示す図

【符号の説明】

101…対応表、102…画面名称、103…JSP名称、104…サーブレットファイル名称、105…Beanファイル名称、107…外部記憶装置、108, 111, 114…設計ドキュメント、109, 112, 115…テンプレート、110, 113, 116…ソースコード。

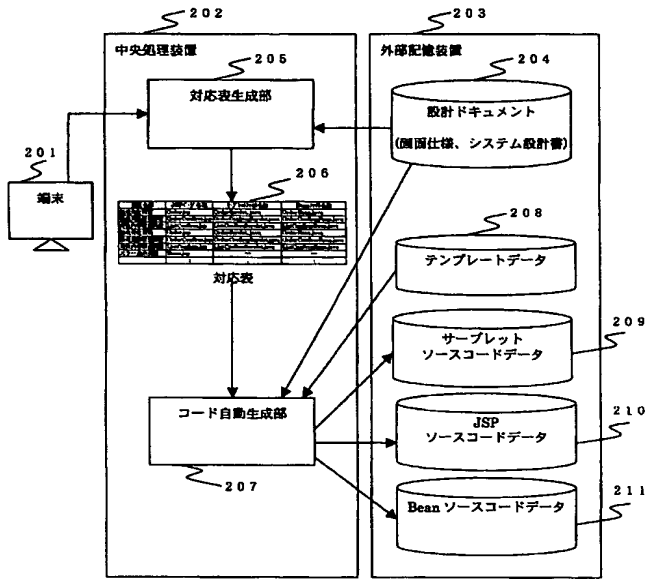
【図1】

対応表



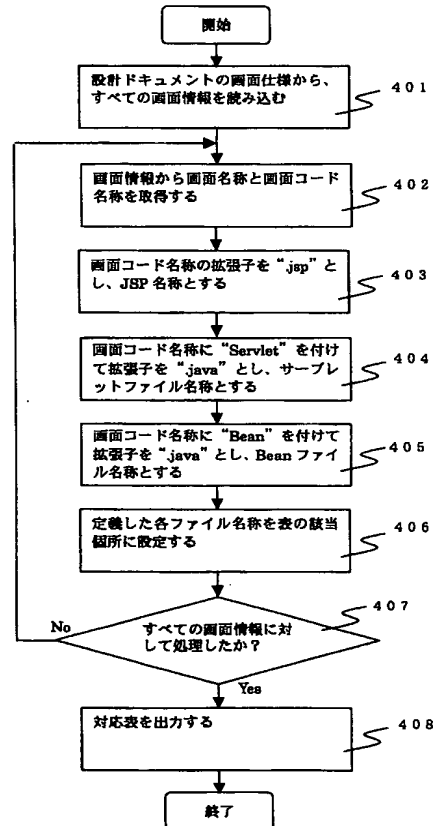
【図2】

システム構成



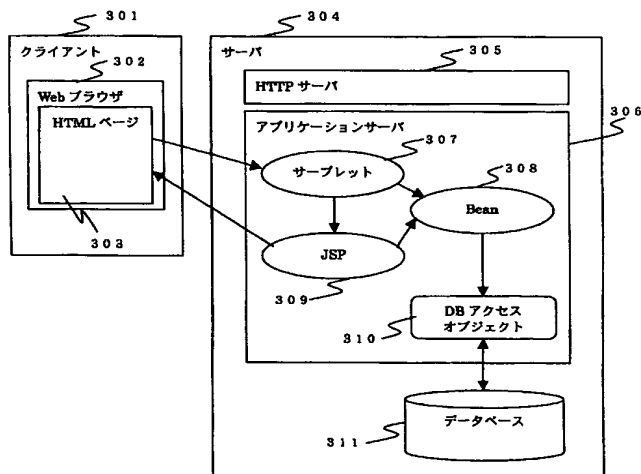
【図4】

対応表の生成処理の流れを示すフローチャート



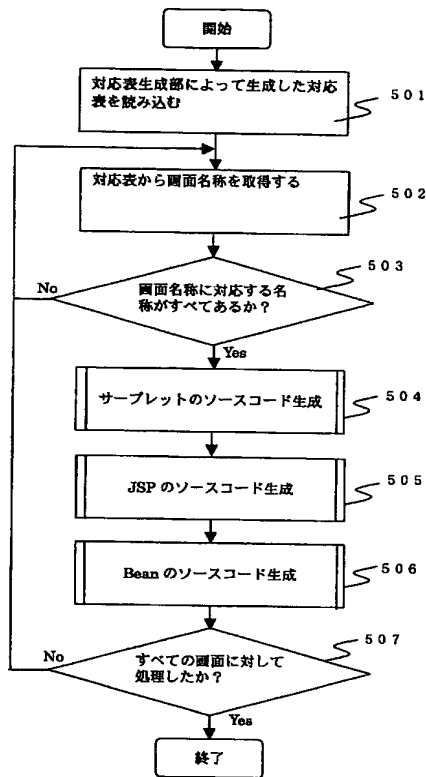
【図3】

サーブレット、JSP ページ、Bean を利用した Web システムの全体構成



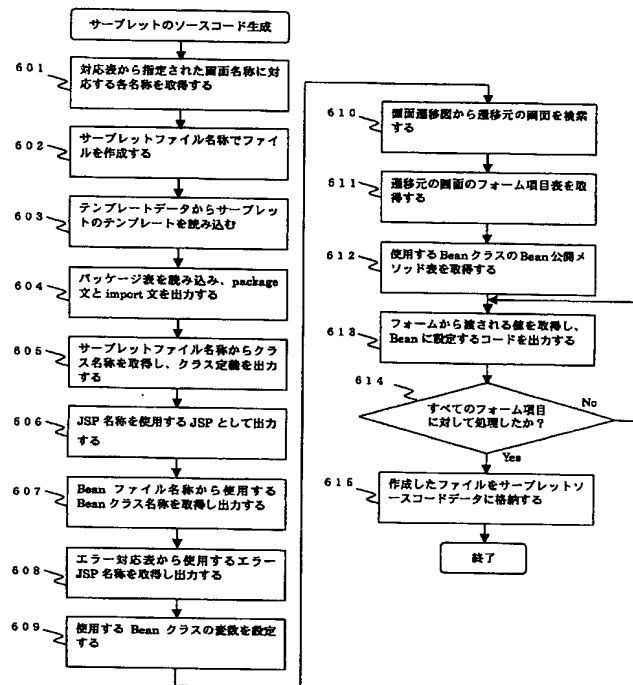
【図5】

ソースコードの自動生成処理の流れを示すフローチャート



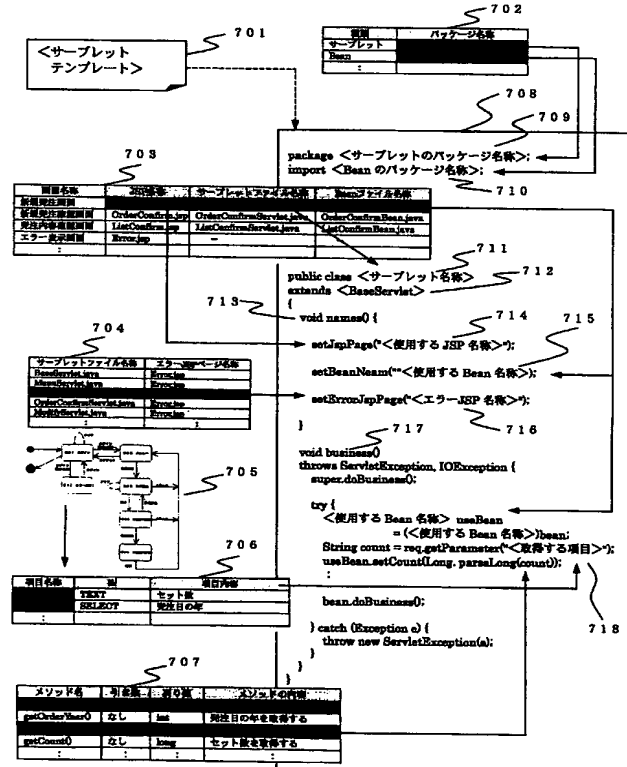
【図6】

サープレットのソースコードを自動生成する処理の流れを示すフローチャート



【図7】

サープレットのソースコード生成



【図 8】

サーブレットのスーパークラスのソースコード

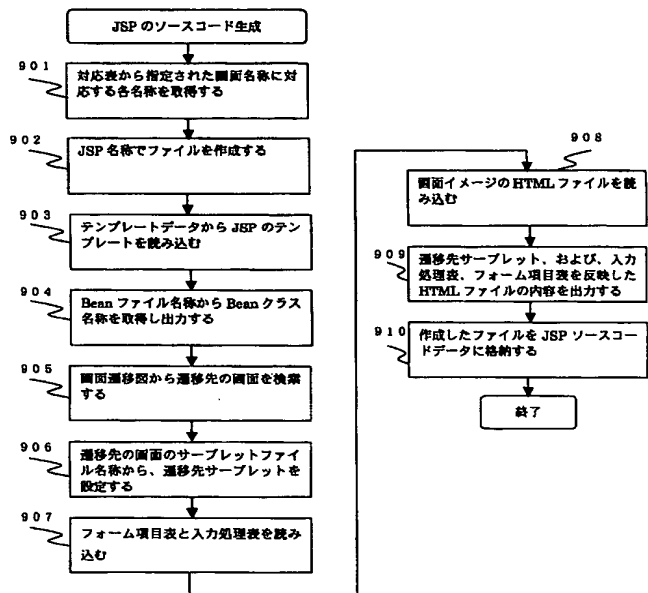
```

package <サーブレットのパッケージ名>;
import <必要なパッケージ>;
abstract public class <BaseServlet> extends HttpServlet {
    String jspPage;           // JSP 名称
    String beanName;          // Bean クラス名称
    String errorJspPage;      // エラー JSP 名称
    <BaseBean> bean;          // 利用する Bean オブジェクト
    void business (HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        bean = (<BaseBean>)Beans.instantiate(this.getClassLoader(), getBeanName());
    }
    void attribute (HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        req.setAttribute("bean", bean);
    }
    void forward (HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        RequestDispatcher rd = getServletContext().getRequestDispatcher(getJspPage());
        rd.forward(req, resp);
    }
    public void init () throws ServletException {
        names ();
    }
    abstract void names ();
    public void doPost (HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        try {
            business(req, resp); // 業務処理
            attribute(req, resp); // bean 設定
            forward(req, resp);  // JSP 呼び出し
        } catch (ServletException e) {
            error (req, resp, e.getMessage());
        }
    }
    void error (HttpServletRequest req, HttpServletResponse resp, String message)
        throws ServletException, IOException {
        setJspPage(getErrorJspPage());
        req.setAttribute("message", message);
        forward();
    }
    void setJspPage (String page) { this.jspPage = page; }
    String getJspPage () { return jspPage; }
    void setBeanName (String name) { this.beanName = name; }
    String getBeanName () { return beanName; }
    void setErrorJspPage (String page) { this.errorJspPage = page; }
    String getErrorJspPage () { return errorJspPage; }
}

```

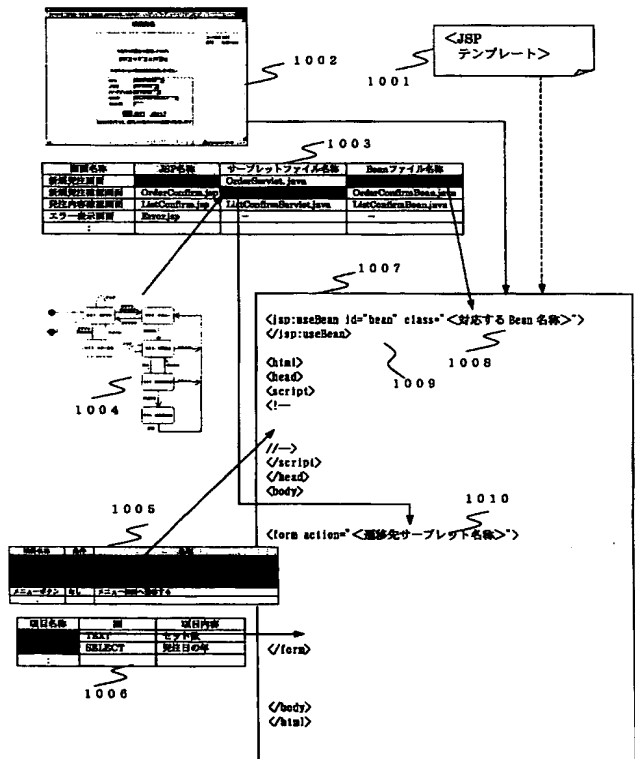
【図 9】

JSP のソースコードを自動生成する処理の流れを示すフローチャート

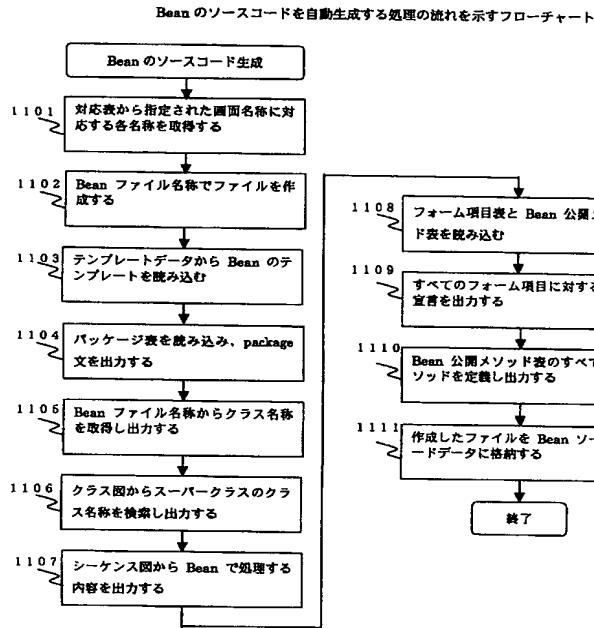


【図 10】

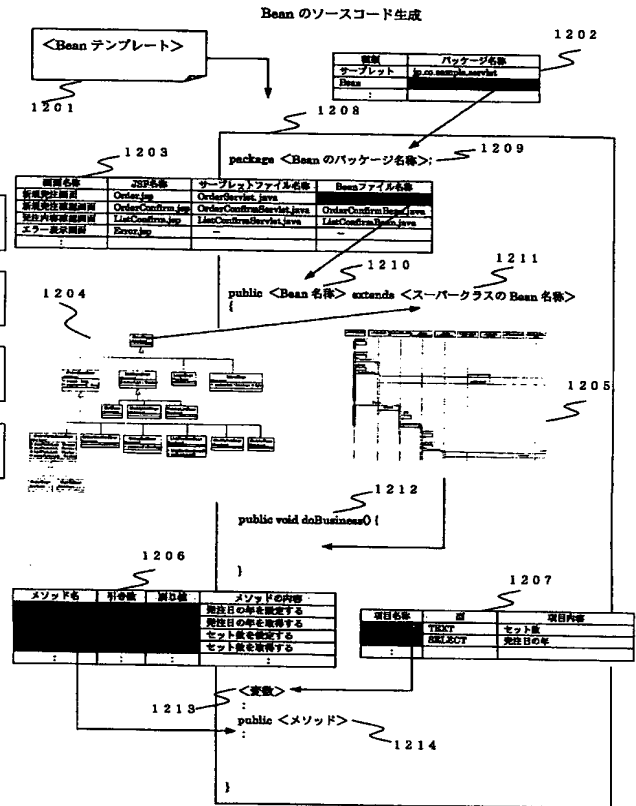
JSP のソースコード生成



【図11】

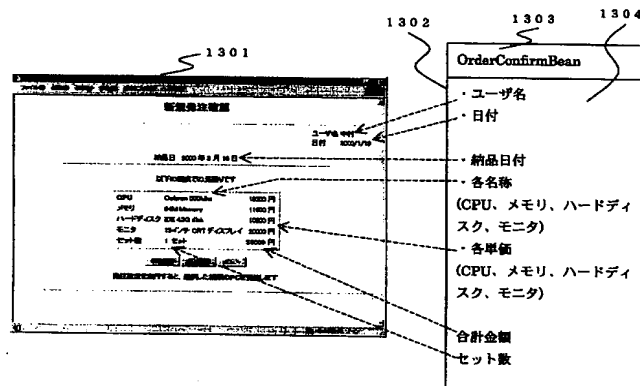


【図12】



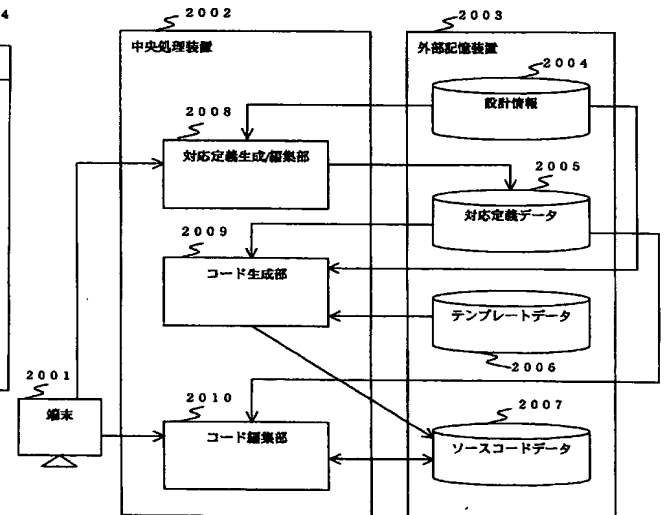
【図13】

画面情報とBeanの関係



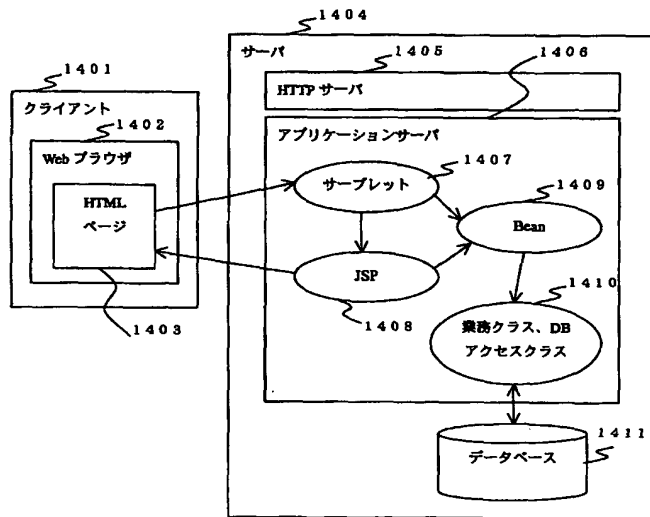
【図20】

システム構成



【図 14】

Servlet と JSP、Bean を利用した Web アプリケーションシステムの全体構成



【図 15】

受信と応答コンポーネントの関係

1501 入力画面	1502 受信コンポーネント	1503 応答コンポーネント	1504 出力画面
画面 A	受信/応答部品 1		画面 B
画面 B	1505 受信窓口部品 1	受信部品 1	画面 C
画面 C		受信部品 2	画面 D
画面 D		受信部品 3	画面 E
画面 E		受信部品 4	画面 F
画面 F		受信部品 5	画面 G
画面 G	受信窓口部品 2	受信部品 6	画面 H
画面 H		応答部品 5	画面 I
画面 I		応答部品 6	画面 J
画面 J	受信部品 7	応答部品 7	画面 K
画面 K	要求部品 8	応答部品 8	画面 L
画面 L	要求部品 9	応答部品 9	画面 M

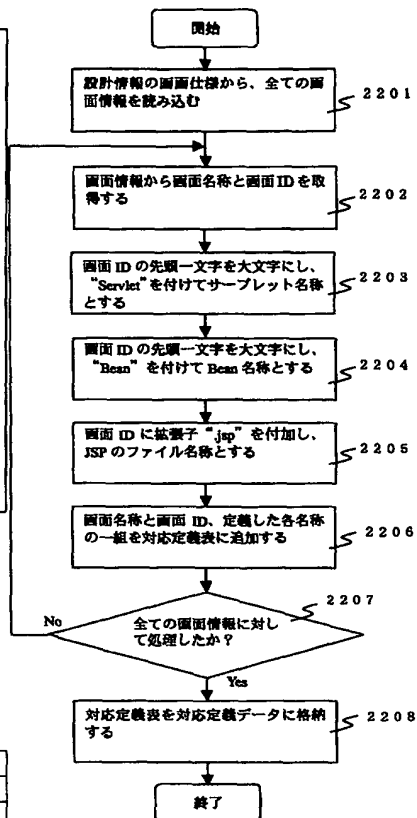
【図 16】

受信と応答コンポーネント定義例 1

1501 入力画面	1601 イベント	1502 受信コンポーネント	1602 条件	1503 応答コンポーネント	1504 出力画面
画面 A	[OK] 押下	Servlet B	なし	JSP B	画面 B
画面 B	[OK] 押下	Servlet C	OK の場合	JSP C	画面 C
画面 C	[OK] 押下	Servlet E	NG の場合	JSP D	画面 D
画面 D	[NG] 押下	Servlet A	なし	JSPA	画面 A

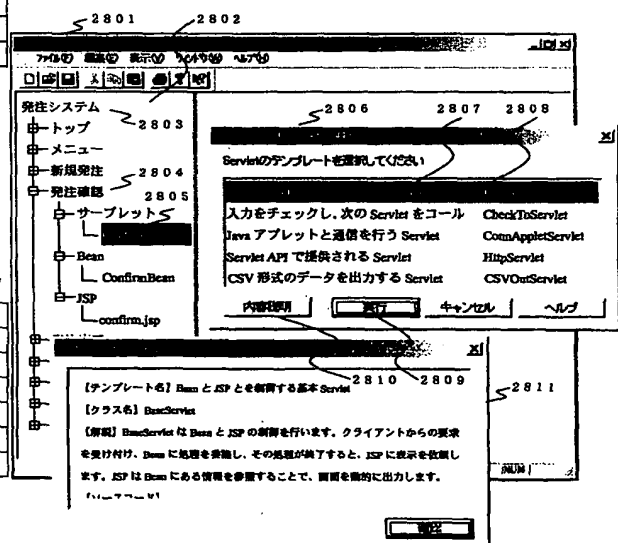
【図 22】

対応定義表の生成処理の流れを示すフローチャート



【図 28】

テンプレートの選択イメージ



【図 17】

受信と応答コンポーネント定義例 2

1501	1502	1601	1602	1503	1504
入力画面	受信コンポーネント	イベント	条件	応答コンポーネント	出力画面
画面 A	ServletA	[OK]押下	なし	JSP B	画面 B
画面 B	ServletB	[OK]押下	OK の場合	JSP C	画面 C
			NG の場合	JSP D	画面 D
画面 C	ServletC	[OK]押下	なし	JSP E	画面 E
画面 D	ServletD	[NG]押下	なし	JSP A	画面 A
画面 E		[OK]押下	なし	JSP A	画面 A

【図 18】

コンポーネントの対応定義表

1801	1802	1803	1804	1805	1806
画面名称	画面ID	サーブレット名称	Bean名称	JSP名称	
トップ	default	DefaultServlet	DefaultBean	default.jsp	
メニュー	menu	MenuServlet	MenuBean	menu.jsp	
新規発注	order	OrderServlet	OrderBean	order.jsp	
新規発注入力不足	orderError			orderError.jsp	
発注確認	confirm	ConfirmServlet	ConfirmBean	confirm.jsp	
			ItemBean		
			PriceBean		
発注完了	complete	CompleteServlet	CompleteBean	complete.jsp	
既発注一覧	list	ListServlet	ListBean	list.jsp	
既発注内容確認	detail	DetailServlet	DetailBean	detail.jsp	
			AmountBean		
発注修正	modify	ModifyServlet	ModifyBean	modify.jsp	
			ItemBean		
発注取り消し	remove	RemoveServlet	RemoveBean	remove.jsp	
ログイン	login	LoginServlet	LoginBean	login.jsp	
ログイン失敗	loginError			loginError.jsp	
エラー表示	error	-	-	error.jsp	

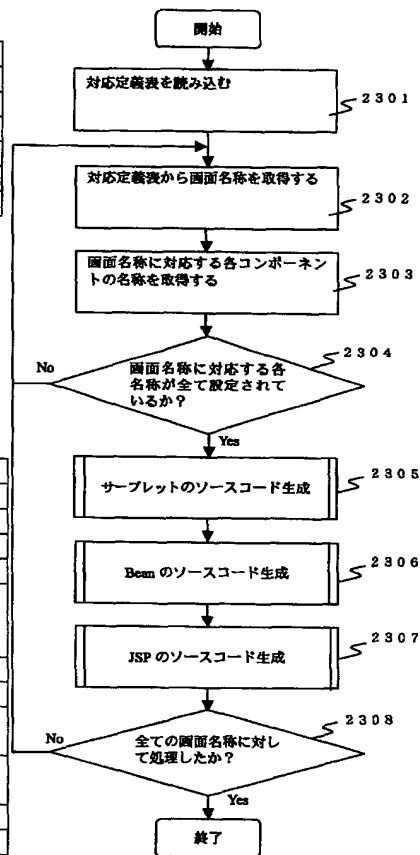
【図 19】

コンポーネントの対応定義表

1901	1902	1903	1904	1905	1906
画面名称	画面ID	サーブレット名称	Bean名称	JSP名称	
トップ	default	DefaultServlet	DefaultBean	default.jsp	
メニュー	menu	MenuServlet	MenuBean	menu.jsp	
新規発注	order	OrderServlet	OrderBean	order.jsp	
新規発注入力不足	orderError			orderError.jsp	
発注確認	confirm	ConfirmServlet	ConfirmBean	confirm.jsp	
発注完了	complete	CompleteServlet	CompleteBean	complete.jsp	
既発注一覧	list	ListServlet	ListBean	list.jsp	
既発注内容確認	detail	DetailServlet	DetailBean	detail.jsp	
発注修正	modify	ModifyServlet	ModifyBean	modify.jsp	
発注取り消し	remove	RemoveServlet	RemoveBean	remove.jsp	
ログイン	login	LoginServlet	LoginBean	login.jsp	
ログイン失敗	loginError			loginError.jsp	
エラー表示	error	-	-	error.jsp	

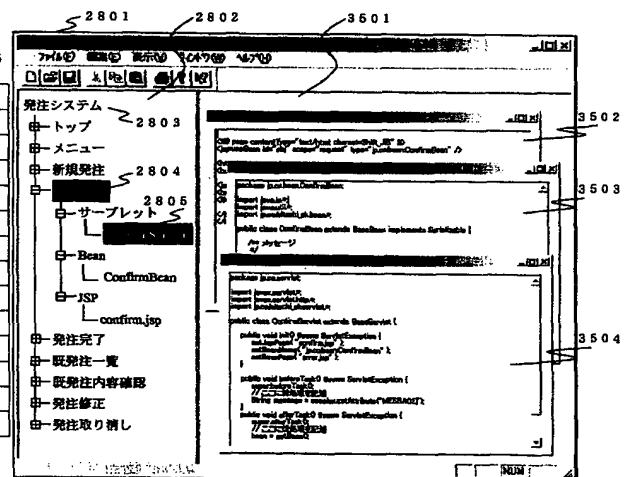
【図 23】

ソースコードの自動生成処理の流れを示すフローチャート

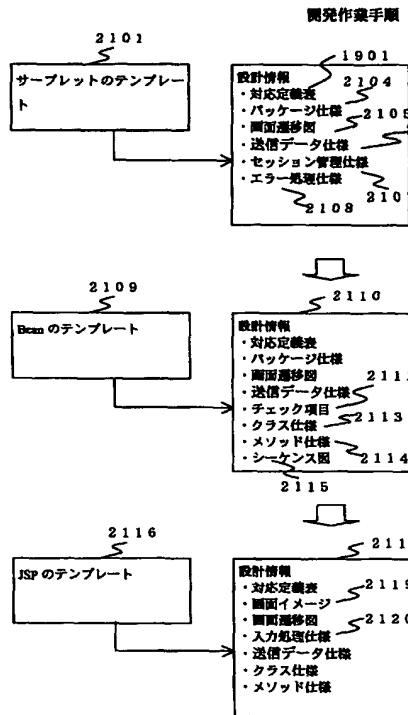


【図 35】

ソースコードの編集イメージ

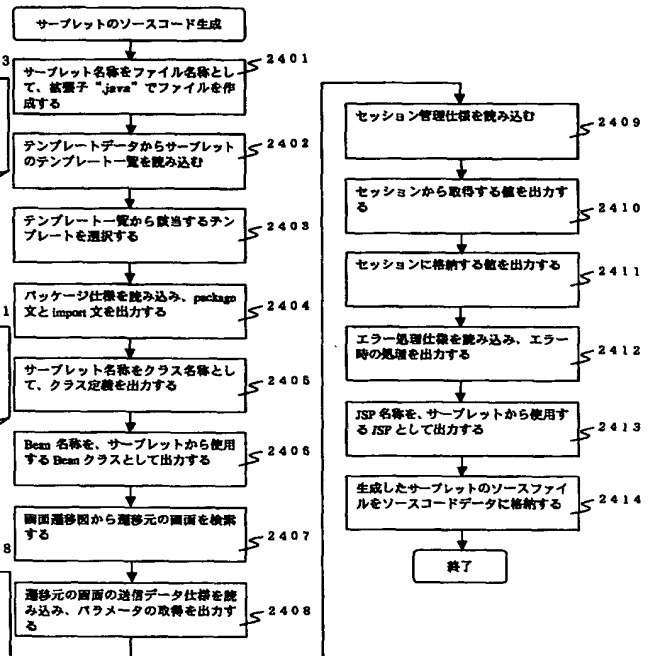


【図 21】



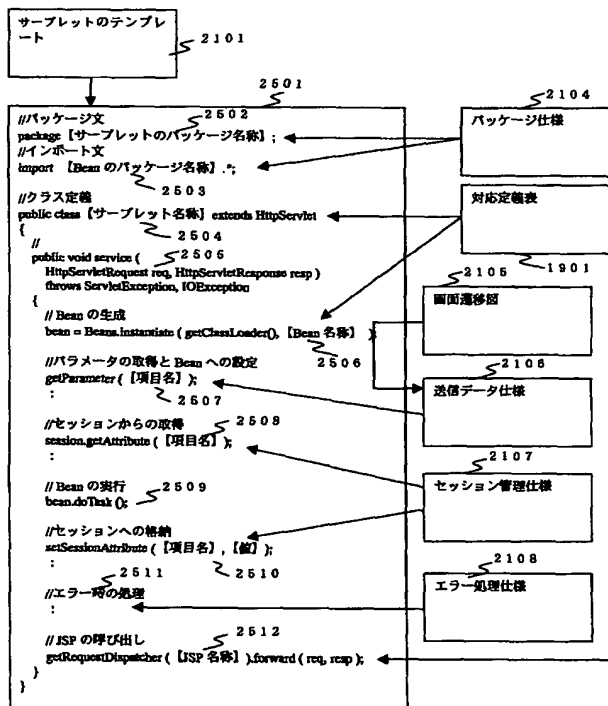
【図 24】

サブリットのソースコードを自動生成する処理の流れを示すフローチャート



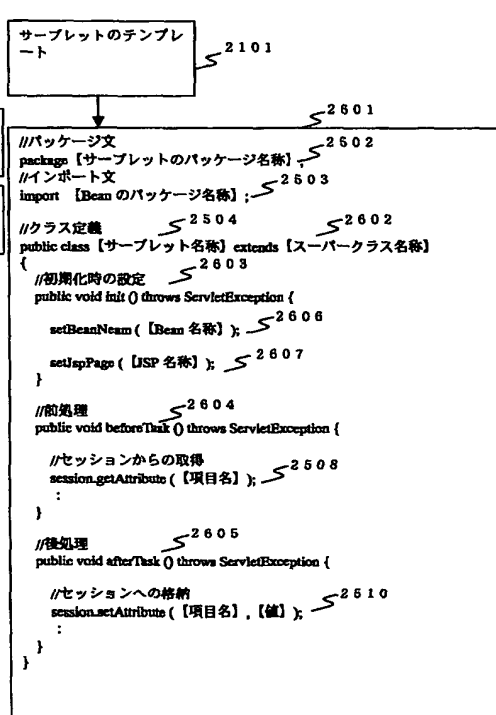
【図 25】

サブリットのソースコード生成例



【図 26】

サブリットのソースコード生成



【図 27】

サーブレットのスーパークラスのコード

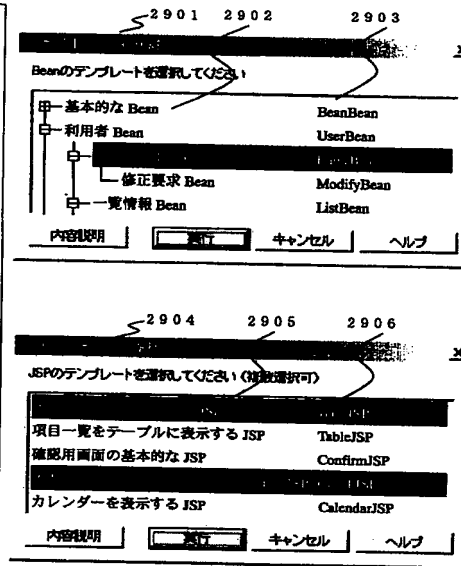
```

2701 public abstract class BaseServlet extends javax.servlet.http.HttpServlet
2702 {
2703     private String jspname;           // JSP 名称
2704     private String beanname;         // Bean 名称
2705     private BaseBean bean;           // Bean オブジェクト
2706
2707     abstract public void init () throws ServletException;
2708
2709     public void service ( HttpServletRequest req, HttpServletResponse resp )
2710     throws ServletException, IOException
2711     {
2712         try {
2713             create ();                // Bean の生成
2714             beforeTask ();             // 前処理
2715             doTask ();                 // Bean の実行
2716             afterTask ();              // 後処理
2717             callJsp ();                // JSP 呼び出し
2718         } catch (Exception e) {
2719             error ();                 // エラー処理
2720         }
2721     }
2722     private void create () throws ServletException {
2723         // Bean を生成する処理
2724     }
2725     protected void beforeTask () throws ServletException {
2726         // 必要であればサブクラスで実装
2727     }
2728     private void doTask () throws ServletException {
2729         // Bean を実行する処理
2730     }
2731     protected void afterTask () throws ServletException {
2732         // 必要であればサブクラスで実装
2733     }
2734     private void callJsp () throws ServletException {
2735         // JSP を呼び出す処理
2736     }
2737     private void error () throws ServletException {
2738         // エラー処理
2739     }
2740     void setJspPage ( String name ) { this.jspname = name; }
2741     String getJspPage () { return jspname; }
2742     void setBeanName ( String name ) { this.beanname = name; }
2743     String getBeanName () { return beanname; }
2744 }

```

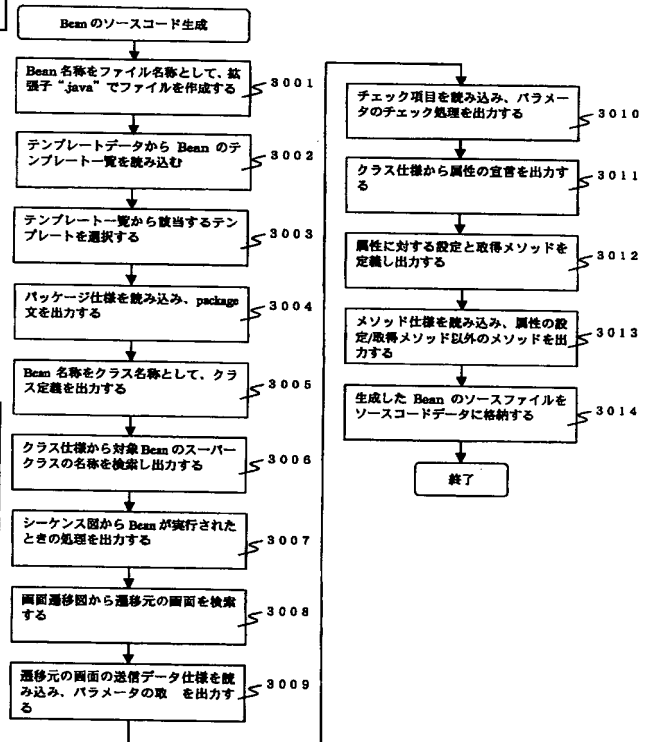
【図 29】

テンプレートの選択イメージ



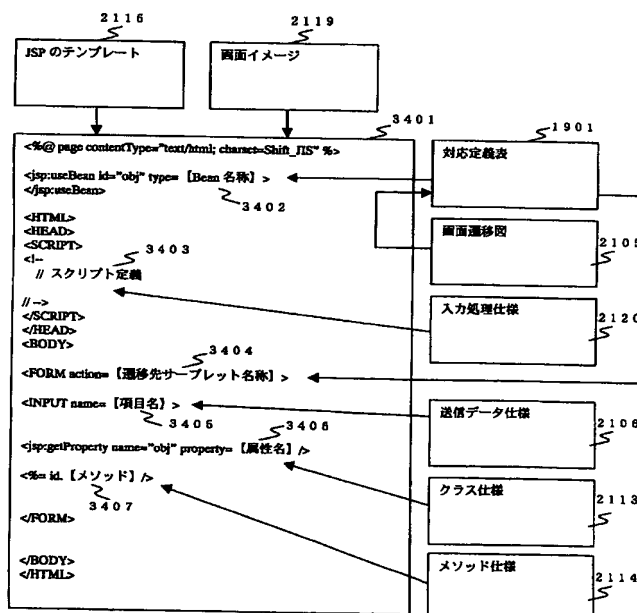
【図 30】

Bean のソースコードを自動生成する処理の流れを示すフローチャート



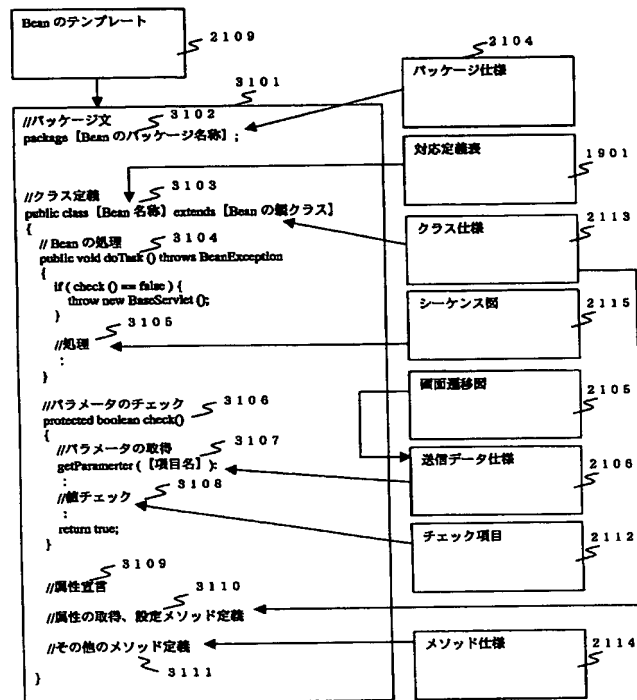
【図 34】

JSP のソースコード生成例



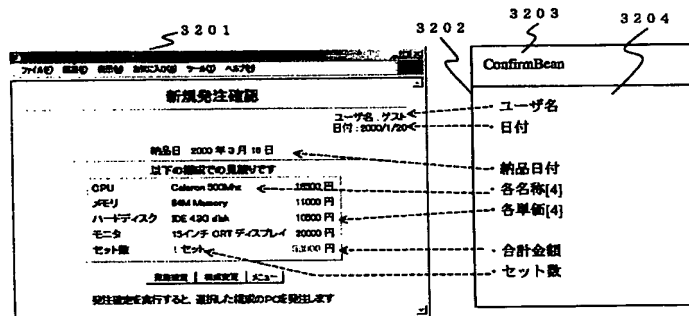
【図31】

Beanのソースコード生成例



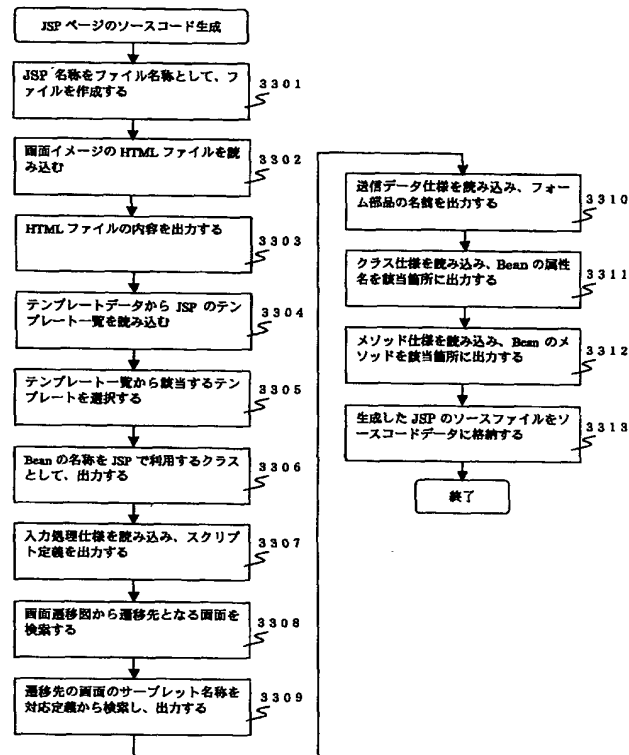
【図32】

画面情報と Bean の属性の関係



【図33】

JSPのソースコードを自動生成する処理の流れを示すフローチャート



フロントページの続き

(72) 発明者 正村 勉
 神奈川県横浜市中区尾上町6丁目81番地
 日立ソフトウェアエンジニアリング株式会社
 社内

(72) 発明者 菅沼 弘
 神奈川県横浜市中区尾上町6丁目81番地
 日立ソフトウェアエンジニアリング株式会社
 社内

(72) 発明者 南部 英俊
 神奈川県横浜市中区尾上町6丁目81番地
 日立ソフトウェアエンジニアリング株式会社
 社内

Fターム(参考) 5B076 DD04 EC07